Basing Cryptographic Protocols on Physical Objects

Basing Cryptographic Protocols on Physical Objects

Daiki Miyahara

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

DOCTOR OF PHILOSOPHY (INFORMATION SCIENCES)

January 2021

Acknowledgements

First, I would like to express my gratitude to my supervisor, Prof. Hideaki Sone, who always supports me and kindly checks my presentations. I learned how to beautifully motivate my work and present it so that even my parents could understand it, which I would like to always keep in my heart. I would also like to express my gratitude to Assoc. Prof. Takaaki Mizuki for his dedicated teaching. He gave me many useful things including the studying and writing skills. Without him, I would not have been to my doctoral course. My special thanks goes to Prof. Hiroki Shizuya, Prof. Xiao Zhou, and Prof. Ayumi Shinohara for being the examiners and their fruitful comments.

I am deeply grateful for Prof. Yu-ichi Hayashi for his attractive way of teaching. I particularly enjoyed skiing at Zao and watching the Omagari fireworks festival planned mainly by him. I am also deeply grateful for Assoc. Prof. Pascal Lafourcade for our longstanding collaboration. Thanks to him as well as his Ph.D. student, Léo Robert, we could explore and establish the area of physical zero-knowledge proof protocols for pencil puzzles. I would like to express my appreciation for Dr. Goichiro Hanaoka, who invited me as being a technical trainee and visiting researcher at AIST. With his financial support, I could present our work at COCOA 2018, in which I was gave many feedbacks from the audience. I would like to express my thanks to Dr. Yuichi Komano. Especially, his enthusiastic support improves on my creating a presentation in PowerPoint. I would also like to express my thanks to Dr. Kazumasa Shinagawa for valuable discussions. He always tries to discover interesting problems and solve them, which impacts my attitude of tracking a research. I would like to appreciate Assoc. Prof. Mitsugu Iwamoto and Asst. Prof. Yohei Watanabe for inventing a workshop for physical cryptography with Dr. Alexander Koch, in which I could spend fruitful time for discussing open problems.

Finally, let me express my deepest gratitude to my former and present members in the Sone–Mizuki laboratory and my invaluable friends in Tohoku University.

Abstract

Security in information communication is based on modern cryptology. One of famous security protocols used daily is Transport Layer Security (TLS), which is necessary when communicating over the Internet. While TLS is actually important, it is difficult for laypeople to understand the principle of TLS because it is automatically performed on computers. This dissertation covers cryptographic ones using everyday objects, such as physical envelopes and a deck of playing cards, unlike the aforementioned modern protocols. These unconventional protocols can be used for not only practical use but also didactic contexts because they can be executed by human hands. Many studies on card-based cryptographic protocols achieving secure multiparty computation (MPC) with a deck of playing cards, i.e., evaluating a predetermined function over private inputs without revealing any information about the inputs more than necessary, have been conducted. Improving the practicality of card-based protocols is important to promote MPC (which has many applications such as analysis of sensitive data) because currently, they are the only easy way to perform MPC with human hands.

In the research on card-based protocols, improving the number of required cards for computing logical functions, such as AND and majority ones, has been mainly studied. For instance, the AND function with two inputs can be computed with (a two colored deck of) six cards and one shuffle, and the lower bounds on the number of required cards have been provided. However, arithmetic circuits should be also efficiently computed, such as comparing two numbers without revealing anything (also known as Yao's Millionaires' problem) for card-based protocols to be more attractive; few studies have focused on this topic. Besides, it has not been studied that some of shuffle actions used in card-based protocols cannot be securely implemented with human hands. If one can completely see through a shuffle action, it means that information about inputs (or outputs) should be leaked, and hence, MPC cannot be realized with a deck of cards. That is, even if card-based protocols can be performed with a small number of cards, they are not practical until they can be securely implemented.

In the first part of this dissertation (Part I), we discuss the aforementioned problems and show that card-based protocols are practical for both the application of computation and implementation. For attractive applications, we propose two card-based implementations of Yao's millionaire protocol with a two-colored deck of cards. One of our implementations is based on the principle behind Yao's millionaire protocol. This implementation requires a cyclic shuffle called the random cut once, and hence, it can

be easily performed. The other one is used to compute the comparison by logical circuits, reducing the number of required cards. Moreover, we extend our implementations to be performed with a commonly available standard deck of cards. For another application, we explore zero-knowledge proof (ZKP) protocols for famous pencil puzzles such as Sudoku. Our studies show that a malicious prover who does not know the solution to a given pencil puzzle can convince a verifier with a probability of zero, i.e., all of our proposed ZKP protocols have no soundness error. In this dissertation, we present card-based ZKP protocols for Sudoku and Slitherlink.

To implement card-based protocols securely, we propose two secure implementations of a random bisection cut, which is a shuffle action of bisecting a sequence of cards and shuffling two halves and is used in many protocols, such as the aforementioned AND protocol. One of our implementations employs a curving polystyrene foam ball split into two halves so that a sequence of cards can be placed in it. The other one is used to simplify the execution of a random bisection cut to a random cut by exploiting the vertical asymmetry of the back of cards. One notable feature of this implementation is that no additional tool is required.

The second part of this dissertation (Part II) describes cryptographic protocols based on everyday tools. We first propose a ball-based protocol, i.e., an MPC protocol using physical balls and bags. For this, we consider the use of an interesting feature in that the balls become disordered once placed in a bag, namely, they are "automatically shuffled." We confirm that our ball-based protocol is based on realistic physical operations and is feasible for humans.

Almost all of the existing physical cryptographic protocols exhibit a major limitation (but not explicitly mentioned previously) in that they cannot be performed remotely. We provide a novel solution to this problem by employing online apps such as Gmail and WhatsApp. In this dissertation, we consider the use of a messaging app, such as Facebook Messenger, and show how the existing secure auction protocol using envelopes can be remotely performed. The read receipts and group chat features play a key role in achieving fairness, privacy, and verifiability in our auction protocol.

To summarize, this thesis significantly improves the practicality of cryptography using everyday objects and contributes to extending them.

Contents

1. Introduction

We communicate with each other today over the Internet, in which a cryptographic protocol called Transport Layer Security (TLS) provides privacy, data integrity, and authentication. TLS consists of combinations of modern cryptology [12] such as symmetric/public-key cryptography, digital signature, and public-key infrastructure. Security in network communication is based on TLS; however, it is difficult for laypeople to understand the principle of TLS because it is automatically performed on computers.

In this thesis, we study cryptographic ones using *physical objects*, such as envelopes and a deck of playing cards, unlike the aforementioned modern protocols. An important aspect of these unconventional protocols is that they employ physical properties, such as invisibility inside an envelope, which is intuitively known and used daily; hence, the correctness and security proof are easily understood (which was discussed in [15, 19, 50, 51]). Therefore, these protocols are used for introducing the notion of cryptography in a university course[i].

Such unconventional protocols include the ones using a PEZ dispenser [1, 4], 15 puzzle [43], dial lock [42], visual secret sharing sheets [10], transparent sheets [35, 75], and coins [33]. Many studies on *card-based cryptographic protocols* [11] have been conducted with a deck of cards to evaluate a predetermined function over private inputs without revealing any unnecessary information regarding the inputs, thus achieving secure multiparty computation (MPC) [83]. MPC has many useful applications such as analysis of sensitive data. Consequently, many studies on improving the efficiency of MPC have been conducted (e.g., [3]); however, MPC generally requires cryptographic techniques such as homomorphic encryption and secret sharing. Therefore, we believe that improving the practicality of card-based protocols is important to promote MPC because currently, they are the only easy way to perform MPC with human hands.

In the research on card-based protocols, improving the number of required cards for computing logical functions, such as AND and majority ones, has been mainly studied. Starting from the most elegant one called the "five-card trick" invented by Den Boer [11] in 1989, card-based protocols for computing logical functions, such as AND and XOR, have been presented [2, 9, 38, 39, 44, 48, 56, 57, 59, 67, 78, 79]. Further, any Boolean function can be efficiently computed with respect to the numbers of required cards [58] and required shuffles [77]. Besides, the lower bounds on the number of required cards for computing the two-input logical AND function

[i] https://www.cs.cornell.edu/courses/cs4830/2008fa/

have been reported with respect to restrictions on running time and practicality of shuffles to be used [16, 26, 27, 29, 31, 63] (which were summarized in a part of the dissertation by Koch [28]). However, card-based protocols should have many applications, such as arithmetic circuits, to be attractive. Typical examples include a secure computation of the comparison between two natural numbers without revealing anything (also known as Yao's Millionaires' problem [83]). A few studies, including the ones based on the card-based millionaires' protocol and secure grouping protocol, have focused on this topic using private operations [52] and using properties of permutations [22]. (Card-based protocols using a tailor-made deck of cards, such as regular polygon cards [76] and dihedral cards [74], were summarized in a part of the dissertation by Shinagawa [73].)

In addition to the applications, the implementation of card-based protocols is yet to be studied. Nakai et al. [52] reported that a random bisection cut invented by Mizuki and Sone [48] (which bisects a sequence of cards, shuffles two halves, and is used in many protocols including the aforementioned AND protocols [44, 48]) requires a private space to be securely implemented, indicating that some of the card-based protocols cannot be executed publicly. Consequently, even if card-based protocols can be performed with a small number of cards and shuffles, they are not practical until they can be securely implemented.

1.1. Contributions and Outline of the Thesis

We divide this dissertation into two parts. Each chapter is self-contained.

In the first part (Part I), we study the aforementioned problems and show that card-based protocols have useful applications and can be securely implemented. In Chapter 2, we propose two card-based implementations of Yao's millionaire protocol with a two-colored deck of cards, such as $\boxed{\clubsuit}$s and $\boxed{\heartsuit}$s. One of our implementations is based on the principle behind Yao's millionaire protocol. This implementation requires a cyclic shuffle called the random cut once, and hence, it can be easily performed. The other one is used to compute the comparison by logical circuits, reducing the number of required cards. Moreover, we extend our implementations to be performed with a commonly available standard deck of cards, such as $\boxed{1}\boxed{2}\boxed{3}\cdots$.

In Chapter 3, we explore *physical zero-knowledge proof* (ZKP) protocols for famous pencil puzzles, such as Sudoku, i.e., we provide a way of convincing a verifier that a prover knows the solution to a puzzle. Our studies show that a malicious prover who does not know the solution to a pencil puzzle can convince a verifier with a probability of zero, i.e., all of our proposed ZKP protocols have no soundness error, improving on the existing work [5, 8, 19]. In this dissertation, we present card-based ZKP protocols for Sudoku and Slitherlink. After our papers for physical ZKP protocols were

published, Ruangwises and Itoh constructed card-based ZKP protocols for other pencil puzzles and graph problems [68–70].

In Chapter 4, we propose two secure implementations of a random bisection cut to implement card-based protocols securely. One of our implementations employs a curving polystyrene foam ball split into two halves so that a sequence of cards can be placed into it. The other one is used to simplify the execution of a random bisection cut to a random cut by exploiting the vertical asymmetry of the back of cards. One notable feature of this implementation is that no additional tool is required.

The second part of this dissertation (Part II) describes cryptographic protocols based on everyday tools. In Chapter 5, we propose an MPC protocol using physical balls and bags, namely, a *ball-based* protocol. For this, we consider the use of an interesting feature in which the balls become disordered once placed in a bag, i.e., they are "automatically shuffled." We confirm that our ball-based protocol is based on realistic physical operations and is feasible for humans.

Almost all the existing cryptographic protocols with physical objects exhibit a major limitation (but not explicitly mentioned previously) in that they cannot be performed remotely. Only cryptographic protocols with tamper-evident seals [50, 51] can perform fair coin-flipping and oblivious transfer by exchanging physical envelopes; however, the required cost and time have not been discussed. In Chapter 6, we provide a novel solution to this issue by employing *online apps*, such as Gmail and WhatsApp. We consider the use of a messaging app, such as Facebook Messenger, and demonstrate how the existing secure auction protocol using envelopes [13] can be remotely performed. The read receipts and group-chat features play a key role in achieving fairness, privacy, and verifiability in our auction protocol.

To summarize, this thesis contributes to the practicality of card-based protocols from two important aspects, namely, applications and implementations, and extends the application of cryptographic protocols using physical objects.

1.2. Publication Overview

Let me summarize how each chapter is organized with reference to our publications.

1.2.1. Part 1: Card-based Cryptographic Protocols

As I mentioned before, Part I contributes to card-based protocols for their applications and implementations.

Chapter 2 is based on the following paper [37]:

Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki

> Sone. "Practical Card-based Implementations of Yao's Millionaire Protocol." *Theoretical Computer Science*, Elsevier, vol.803, pp.207–221, 2020. DOI: j.tcs.2019.11.005. © 2019 Elsevier B.V.

In addition to the content of the above paper, we consider and solve a variant of Yao's millionaire problem.

Chapter 3 is based on the following two papers, to which I contributed in the contructions of the proposed protocols and the papers [34, 71]:

> Tatsuya Sasaki, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. "Efficient Card-Based Zero-Knowledge Proof for Sudoku." *Theoretical Computer Science*, Elsevier, vol.839, pp.135–142, 2020. DOI: j.tcs.2020.05.036. © 2020 The Authors. Published by Elsevier B.V.

> Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Tatsuya Sasaki, and Hideaki Sone. "A Physical ZKP for Slitherlink: How to Perform Physical Topology-preserving computation". *15th International Conference on Information Security Practice and Experience* (ISPEC 2019), Lecture Notes in Computer Science, Springer, vol.11879, pp.135–151, 2019. DOI: 10.1007/978-3-030-34339-2_8. © Springer Nature Switzerland AG 2019.

Moreover, we discuss a lower boud on the number of required cards for card-based ZKP protocols for Sudoku.

Chapter 4 is based on the following paper, to which I contributed in using the vertical asymmetricity of cards, experimenting a more secure way of applying the random bisection cut, and the construction of the paper [80]:

> Itaru Ueda, Daiki Miyahara, Akihiro Nishimura, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Secure Implementations of a Random Bisection Cut." *International Journal of Information Security*, Springer, vol.19, pp.445–452, 2020. DOI: 10.1007/s10207-019-00463-w. © Springer-Verlag GmbH Germany, part of Springer Nature 2019.

1.2.2. Part 2: Cryptographic Protocols with Everyday Tools

As I mentioned before, Part II explores the area of cryptographic protocols with everyday tools.

Chapter 5 is based on the following paper:

> Daiki Miyahara, Yuichi Komano, Takaaki Mizuki, and Hideaki Sone. "Cooking Cryptographers: Secure Multiparty Computation Based on Balls and Bags." in submission.

Chapter 6 is based on the following paper:

Daiki Miyahara, Yuichi Komano, Takaaki Mizuki, and Hideaki Sone. "How to Perform Physical Auction Protocol Remotely with Messaging App." In the proceedings of Computer Security Symposium 2020 (CSS2020), Online, October 26–29, 2020, (in Japanese). © 2020 CSS2020 Organizing Committee.

Part I.

Card-based Cryptographic Protocols

2. Millionaires' Problem

Assume that Alice and Bob have a and b dollars, respectively, such that $a, b \in \{1, 2, \ldots, m\}$ for some natural number m. They want to know who is richer without revealing any information about their values (more than that is necessary), i.e., they want to determine only whether $a < b$ or not. This is the famous *millionaires' problem* proposed by Yao [83] in 1982, and he designed a protocol, which we call *Yao's millionaire protocol*, to solve the problem based on a public-key cryptosystem. The fundamental principle behind Yao's millionaire protocol could be interpreted as follows. If Alice arranges m symbols consisting of a number a of ♠s and a number $(m-a)$ of ♢s as

$$\overset{1}{\spadesuit}\ \overset{2}{\spadesuit}\ \cdots\ \overset{a}{\spadesuit}\ \overset{a+1}{\diamondsuit}\ \overset{a+2}{\diamondsuit}\ \cdots\ \overset{m}{\diamondsuit},$$

and Bob points at the b-th symbol, then the b-th symbol being ♢ implies $a < b$, and the b-th symbol being ♠ implies $a \geq b$:

$$\overset{1}{\spadesuit}\ \overset{2}{\spadesuit}\ \cdots\ \overset{a}{\spadesuit}\ \overset{a+1}{\diamondsuit}\ \overset{a+2}{\diamondsuit}\ \cdots\ \underset{\substack{\uparrow\\ b\text{-th}}}{\overset{b}{\diamondsuit}}\ \cdots\ \overset{m}{\diamondsuit} \iff a < b\,,$$

$$\overset{1}{\spadesuit}\ \overset{2}{\spadesuit}\ \cdots\ \underset{\substack{\uparrow\\ b\text{-th}}}{\overset{b}{\spadesuit}}\ \cdots\ \overset{a}{\spadesuit}\ \overset{a+1}{\diamondsuit}\ \overset{a+2}{\diamondsuit}\ \cdots\ \overset{m}{\diamondsuit} \iff a \geq b\,.$$

While Yao's millionaire protocol relies on the public-key cryptosystem to implement the above principle without leaking actual values a and b, Nakai, Tokushige, Misawa, Iwamoto, and Ohta [52] considered the use of a deck of physical cards in 2016. That is, following the fundamental principle above, they constructed a card-based scheme using cards of two types such as

$$\boxed{\clubsuit}\,\boxed{\clubsuit}\ \cdots\ \boxed{\clubsuit}\,\boxed{\heartsuit}\,\boxed{\heartsuit}\ \cdots\ \boxed{\heartsuit}$$

whose backs are all identical $\boxed{?}$. Roughly speaking, in their scheme, Alice first encodes her secret value a with a sequence of face-down cards, and then Bob "privately" changes the positions of cards according to his secret value b. We will describe the details in Section 2.1. Since many people on earth are familiar with playing cards, their card-based scheme is human-friendly and useful. Its only drawback is that it requires a player's "private" action, called *Private Permutation* (PP) [52], which permits Bob to rearrange the sequence of cards privately (for example, he is allowed to manipulate the cards behind his back). Private Permutation is considered to be such a strong assumption that a malicious player may do an active attack. Hereinafter, we refer to their scheme as the *NTMIO protocol with PP*; the acronym NTMIO is made of the initial letters of the names of the authors [52].

Table 2.1: The PP-free millionaire protocols

	Deck	#Cards	#Shuffles	Section
Our implementation with RC	Two-colored	$3m+1$	1	§ 2.2
Using a standard deck	Standard	$4m$	4	§ 2.3
The previous circuit-based [52]	Two-colored	$4\lceil \log m \rceil + 4$	$7\lceil \log m \rceil - 6$	§ 2.4
Our improved circuit-based	Two-colored	$4\lceil \log m \rceil + 2$	$2\lceil \log m \rceil - 1$	§ 2.5

Contribution

Thus, it is preferable to construct a card-based scheme which does not rely on Private Permutation, in order to avoid possible malicious actions. To this end, in this chapter, we present a "PP-free" scheme, which implements the fundamental principle behind Yao's millionaire protocol; instead of using Private Permutation, we use a familiar shuffling operation called the *random cut* (RC). A random cut is a cyclic shuffle, which can be easily implemented by humans as in the case of usual card games (e.g. [11,30,81]). Therefore, our scheme, named the *PP-free protocol with RC*, can be conducted completely publicly, and hence, any malicious action can be detected. As will be seen in Section 2.2, we straightforwardly implement the above principle. Therefore, we believe that even non-experts can easily understand the correctness and secrecy of our scheme, and can practically use it in everyday life.

Similarly to the NTMIO protocol with PP, our PP-free protocol with RC uses a two-colored deck of cards $\clubsuit\clubsuit\cdots\clubsuit\heartsuit\heartsuit\cdots\heartsuit$. Compared with such a two-colored deck of cards, a standard deck of playing cards is more familiar with us. Therefore, we extend our protocol so that we can solve the millionaire problem using a standard deck of playing cards (which is sold in many toy stores all over the world). We note that simply replacing a two-colored deck with a standard deck in the PP-free protocol with RC does not work. Note also that our protocol uses another simple shuffle aside from RC. These results will be presented in Section 2.3.

It should be noted that Nakai et al. [52] proposed a PP-free protocol as well; they presented a card-based scheme, which follows not the above-mentioned fundamental principle but a logical circuit representing the comparison $a < b$. This PP-free circuit-based protocol relies on a shuffling operation called the random bisection cut [48] (instead of Private Permutation). In this chapter, we improve upon this existing protocol; we will reduce the number of required random bisection cuts to around 2/7. We will explain the details in Sections 2.4 and 2.5.

Table 2.1 summarizes the performance of the PP-free protocols.

Outline

The remainder of this chapter is organized as follows. In Section 2.1, we introduce the NTMIO protocol with PP [52]. In Section 2.2, we present our implementation, the PP-free protocol with RC. We confirm that simply replacing a two-colored deck with a standard deck in the PP-free protocol with RC cannot solve the millionaire problem, and then present how to resolve the issue in Section 2.3. As for circuit-based protocols, we introduce the previous protocol in Section 2.4, and give an improved protocol in Section 2.5. Moreover, for the circuit-based protocols, we consider the use of a standard deck of cards in Section 2.6. We conclude this chapter in Section 2.7.

2.1. The Previous Scheme: NTMIO Protocol with PP

In this section, we introduce the NTMIO protocol with PP [52].

Recall the fundamental principle behind Yao's millionaire protocol; Alice arranges m symbols:

$$\overset{1}{\spadesuit}\ \overset{2}{\spadesuit}\ \cdots\ \overset{a}{\spadesuit}\ \overset{a+1}{\diamondsuit}\ \overset{a+2}{\diamondsuit}\ \cdots\ \overset{m}{\diamondsuit}.$$

Using a pair of physical cards $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, let us encode each symbol as follows:

$$\boxed{\clubsuit}\boxed{\heartsuit} = \spadesuit, \quad \boxed{\heartsuit}\boxed{\clubsuit} = \diamondsuit.$$

Thus, Alice can encode her private value a using m pairs of $\boxed{\clubsuit}\boxed{\heartsuit}$, and put the cards with their faces down such that Bob does not see the order of the cards. For such a sequence of m pairs encoding Alice's secret value a, Bob needs to point at the b-th pair without leaking any information about his secret value b; to this end, Bob is permitted to use Private Permutation. Specifically, the NTMIO protocol with PP proceeds as follows.

1. Alice holding m $\boxed{\clubsuit}$s and m $\boxed{\heartsuit}$s places a number a of $\boxed{\clubsuit}\boxed{\heartsuit}$s on a table with their faces down, and then puts $(m-a)$ $\boxed{\heartsuit}\boxed{\clubsuit}$s next to them:

$$\overset{1}{\boxed{?}\boxed{?}}_{\clubsuit\heartsuit}\ \overset{2}{\boxed{?}\boxed{?}}_{\clubsuit\heartsuit}\cdots\overset{a}{\boxed{?}\boxed{?}}_{\clubsuit\heartsuit}\ \overset{a+1}{\boxed{?}\boxed{?}}_{\heartsuit\clubsuit}\ \overset{a+2}{\boxed{?}\boxed{?}}_{\heartsuit\clubsuit}\cdots\overset{m}{\boxed{?}\boxed{?}}_{\heartsuit\clubsuit},$$

   while Bob does not see the order of each pair.

2. Bob uses Private Permutation; he takes the sequence of cards and move them behind his back. Then, he moves the b-th pair to the first without Alice seeing which pair comes first:

$$\begin{array}{cc} & \overset{1}{\boxed{?}\boxed{?}}\cdots\overset{b-1}{\boxed{?}\boxed{?}}\ \overset{b}{\boxed{?}\boxed{?}}\ \overset{b+1}{\boxed{?}\boxed{?}}\cdots\overset{m}{\boxed{?}\boxed{?}} \\ \rightarrow & \overset{b}{\boxed{?}\boxed{?}}\ \overset{1}{\boxed{?}\boxed{?}}\cdots\overset{b-1}{\boxed{?}\boxed{?}}\ \overset{b+1}{\boxed{?}\boxed{?}}\cdots\overset{m}{\boxed{?}\boxed{?}}. \end{array}$$

3. The first pair of cards is revealed.
   - If the revealed cards are $\boxed{\heartsuit}\boxed{\clubsuit}$, $a < b$.
   - If the revealed cards are $\boxed{\clubsuit}\boxed{\heartsuit}$, $a \geq b$.

This is the existing card-based solution to the millionaires' problem using Private Permutation[i]. Let us stress that Bob needs to use Private Permutation in Step 2.

The use of Private Permutation is so powerful as to contribute to improving the efficiency of card-based protocols [52, 53, 64–66, 82], and also it is used in other physical secure protocols [4, 43]; however, it might lead to some issues. To implement Step 2 of this protocol, the following issues are considered. (1) If Bob were malicious, he could make an active attack; for instance, he could replace the sequence of cards with another set of cards (prepared by himself beforehand) behind his back so that he would be able to peep the exact value of a later; because this attack is done behind his back, Alice does not notice it. (2) Alice and/or audience watching the execution of the protocol could learn Bob's secret value b by observing his tiny shoulder movement. (3) Permuting some cards behind one's back might be challenging because one only has to rely on the sense of hands; the case of $b = 1$ or $b = m$ might be no problem, but if $b = m/2$, Bob might have difficulty in searching the desired pair of cards.

In the next section, we design a simple PP-free protocol.

2.2. Our Implementation Using a Random Cut

In this section, we present our card-based implementation of Yao's millionaire protocol; instead of relying on Private Permutation, we use

- a random cut (RC), which is a well-known and easy-to-perform shuffle, and
- cards whose backs are $\boxed{\#}$, which is a different pattern from $\boxed{?}$.

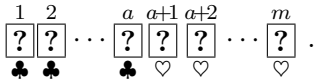
2.2.1. How to Proceed

Our PP-free protocol with RC proceeds as follows.

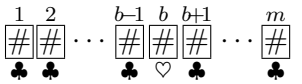
1. Alice holds m $\boxed{\clubsuit}$s and $(m-1)$ $\boxed{\heartsuit}$s. Depending on her secret value a, she places a number a of $\boxed{\clubsuit}$s on a table with their faces down, and then

[i] It should be noted that Fagin, Naor, and Winkler proposed a similar idea to solve the socialist millionaires' problem [25] where Alice and Bob want to know whether they think the same person in mind or not (see Solution 11 in [15]). In addition, Nakai et al. [52] presented another card-based scheme with Private Permutation, which compares a and b bit by bit with the help of "storage" cards.

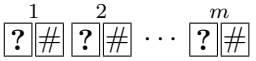
puts a number $(m-a)$ of $\boxed{\heartsuit}$s next to them. The resulting sequence is Alice's input:



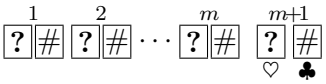
On the other hand, Bob holds $(m-1)$ cards of ♣ whose backs are $\boxed{\#}$ and a card of ♡ whose back is also $\boxed{\#}$. Then, he places these m cards with their faces down on the table such that only the b-th card is ♡. The resulting sequence is Bob's input:



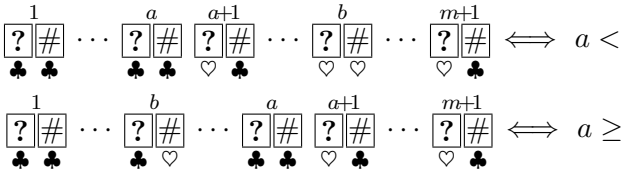
2. Take every card from Alice's input sequence and Bob's input sequence from the left alternately one by one, and put it to the right of the previous card:



We further add two cards to the sequence:

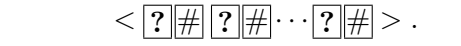


these two cards are put for handling the case of $a = b = m$. Note that recalling the fundamental principle behind Yao's millionaire protocol, the left card of Bob's ♡-card determines whether $a < b$ or not:



Note, furthermore, that when $a \geq b$, the $(b+1)$-st pair determines whether $a = b$ or $a > b$: if the $(b+1)$-st pair is $\boxed{\heartsuit}\boxed{\clubsuit}$ then $a = b$; if it is $\boxed{\clubsuit}\boxed{\clubsuit}$ then $a > b$. Of course, we cannot open Bob's cards $\boxed{\#}$ now; hence, we add a randomization in the next step.

3. Apply a random cut to the sequence of $(2m+2)$ cards, which means shuffling the card sequence cyclically (we denote this operation by $<\cdot>$):



The random cut can be securely implemented by the shuffle operation called the "Hindu cut" [81]; the shuffle may be repeated by Alice and Bob, or even other people until they are all satisfied with the result. Note that the random cut can be done completely publicly [81], and hence, each player can notice any illegal action if any.

4. Reveal all the cards whose backs are $\boxed{\#}$ (namely, the m cards placed by Bob and the additional card); then, one card of ♡ appears. Reveal the card on its left.
   - If the revealed card is $\boxed{\heartsuit}$, $a < b$.
   - If the revealed card is $\boxed{\clubsuit}$, we have $a \geq b$. To see whether equality holds or not, open the card to the right of Bob's ♡-card (apart from cyclic rotation). If the opened card is $\boxed{\heartsuit}$, $a = b$. If it is $\boxed{\clubsuit}$, $a > b$.

This is our PP-free protocol with RC. It uses $(3m+1)$ cards in total and uses one shuffle. In Step 1, Alice places a $\boxed{\clubsuit}$s; if Alice has only a $\boxed{\clubsuit}$s at first, the value a might be leaked from the number of cards that Alice holds. Therefore, Alice needs to have m $\boxed{\clubsuit}$s at first (the number of $\boxed{\heartsuit}$ is similar). Since we apply a random cut in Step 3, revealing Bob's cards in Step 4 does not expose where Bob placed the ♡-card. If $a = b$, Alice and Bob will learn the exact value; note that their values are not leaked to any other people watching the execution of the protocol.

As for the use of a different back $\boxed{\#}$, we were inspired by the technique called the "Chosen Cut" that Koch and Walzer proposed [30][ii]. If the back-side symbol of the cards is vertically asymmetric, we do not need cards of different backs like $\boxed{\#}$: It suffices that Bob puts his cards upside down as follows:

$$\boxed{?}\boxed{\text{ʖ}}\boxed{?}\boxed{\text{ʖ}}\cdots\boxed{?}\boxed{\text{ʖ}}.$$

Our protocol can be executed completely publicly. Any malicious action will be noticed. Moreover, we can automatically confirm that Bob put his input in a correct format when we reveal all Bob's cards in Step 4. We can even be convinced that Alice put her input in a correct format by applying the idea in [46] with some additional cards.

2.2.2. Pseudocode

In this subsection, we present a more formal description of our protocol, that is, we show a pseudocode that follows the computational model of card-based protocols, which was formalized in [31, 45, 47].

First, let us describe an input card sequence. Remember that, for example, if $a = b = 1$, then Alice and Bob will arrange their inputs with two additional cards as:

$$\Gamma^{(1,1)} = (\overbrace{\tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \ldots, \tfrac{?}{\heartsuit}}^{m \text{ cards}}, \overbrace{\tfrac{\#}{\heartsuit}, \tfrac{\#}{\clubsuit}, \ldots, \tfrac{\#}{\clubsuit}}^{m \text{ cards}}, \tfrac{?}{\heartsuit}, \tfrac{\#}{\clubsuit}).$$

[ii] Koch and Walzer [30] showed that one can securely "choose" a permutation from a specific set using helping cards with a different color.

Generally, for $a, b \in \{1, 2, \ldots, m\}$, we define

$$\Gamma^{(a,b)} = (\overset{1}{\underset{\clubsuit}{?}}, \ldots, \overset{a-1}{\underset{\clubsuit}{?}}, \overset{a}{\underset{\clubsuit}{?}}, \overset{a+1}{\underset{\heartsuit}{?}}, \ldots, \overset{m}{\underset{\heartsuit}{?}}, \overset{m+1}{\underset{\clubsuit}{\#}}, \ldots, \overset{m+b-1}{\underset{\clubsuit}{\#}}, \overset{m+b}{\underset{\heartsuit}{\#}}, \overset{m+b+1}{\underset{\clubsuit}{\#}}, \ldots, \overset{2m}{\underset{\clubsuit}{\#}}, \overset{2m+1}{\underset{\heartsuit}{?}}, \overset{2m+2}{\underset{\clubsuit}{\#}}).$$

Next, we need to define the following operations applied to a card sequence $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_d)$:

- (turn, T) for $T \subseteq \{1, 2, \ldots, d\}$, i.e., turning over cards is denoted by a set T such that every card whose position in T is turned over;
- (perm, π) for $\pi \in S_d$, where S_i denotes the symmetric group of degree i, i.e., a rearranging operation is denoted by permutation π;
- $(\mathsf{shuf}, \Pi, \mathcal{F})$ for $\Pi \subseteq S_d$ and a probability distribution $\mathcal{F}$ on Π, i.e., a shuffling operation is denoted by a permutation set Π and a probability distribution $\mathcal{F}$ on Π. If $\mathcal{F}$ is uniform, we simply write it as (shuf, Π);
- (result, e) for some expression e. This indicates that the protocol terminates with the output e.

Based on the above formalization, a pseudocode of our PP-free protocol with RC is shown in Protocol 1, where "visible seq." denotes what we can look at for a card sequence on the table, and we define

$$\sigma \coloneqq \begin{pmatrix} 1 & 2 & 3 & \cdots & m & m{+}1 & m{+}2 & \cdots & 2m & 2m{+}1 & 2m{+}2 \\ 1 & 3 & 5 & \cdots & 2m{-}1 & 2 & 4 & \cdots & 2m & 2m{+}1 & 2m{+}2 \end{pmatrix},$$

which corresponds to the action for taking cards alternately in Step 2 of the protocol, and

$$\mathsf{RC}_{2m+2} \coloneqq \{(1\ 2\ 3\ \cdots\ 2m{+}2)^j \mid 1 \leq j \leq 2m{+}2\},$$

where $(1\ 2\ 3\ \cdots\ 2m{+}2)$ is a cyclic permutation, meaning that $1 \mapsto 2$, $2 \mapsto 3$, and so on.

2.2.3. Example of Real Execution

Our protocol is quite simple and easy-to-implement. For example, two colleagues, Alice and Bob, in a company are easily able to compare their bonuses by using our protocol, where Alice's bonus is 10^a dollars and Bob's bonus is 10^b dollars. The protocol falls into real world cryptography; Fig. 2.1 shows a real execution of our protocol for $m = 4$, i.e., $10^a, 10^b \in \{\$10, \$100, \$1000, \$10000\}$, requiring only 13 cards.[iii] Card-based protocols are far more practical than might be imagined.

[iii] Remember that the protocol requires $3m + 1$ (=13) cards to allow Alice to input a such that $1 \leq a \leq m$ (although there are only 10 cards on the table in Fig. 2.1).

Protocol 1. *The PP-free protocol with RC*

input set: $\left\{\Gamma^{(a,b)} \mid 1 \le a, b \le m\right\}$

Steps:

1. (perm, σ)
2. $(\mathsf{shuf}, \mathsf{RC}_{2m+2})$
3. **if** visible seq. $= (\#, ?, \#, ?, \ldots, \#, ?)$ **then**
4. $\quad (\mathsf{perm}, (2m{+}2 \;\; 2m{+}1 \cdots 1))$
5. $(\mathsf{turn}, \{2, 4, \ldots, 2m{+}2\})$
6. **let** r **s.t.** visible seq. $= (\overbrace{?,\clubsuit}^{1\text{st}}, \ldots, \overbrace{?,\clubsuit}^{(r-1)\text{-st}}, \overbrace{?,\heartsuit}^{r\text{-th}}, \overbrace{?,\clubsuit}^{(r+1)\text{-st}}, \ldots, \overbrace{?,\clubsuit}^{(m+1)\text{-st}})$
7. $(\mathsf{turn}, \{2r{-}1\})$
8. **if** visible seq. $= (?,\clubsuit, \ldots, \overbrace{\heartsuit,\heartsuit}^{r\text{-th}}, \ldots, ?,\clubsuit)$ **then** $(\mathsf{result}, \text{“}a < b\text{”})$
9. **else if** visible seq. $= (?,\clubsuit, \ldots, \overbrace{\clubsuit,\heartsuit}^{r\text{-th}}, \ldots, ?,\clubsuit)$ **then**
10. $\quad (\mathsf{turn}, \{2r{+}1 \pmod{2m{+}2}\})$
11. $\quad$ **if** visible seq. $= (?,\clubsuit, \ldots, \overbrace{\clubsuit,\heartsuit}^{r\text{-th}}, \overbrace{\heartsuit,\clubsuit}^{(r+1)\text{-st}}, \ldots, ?,\clubsuit)$ **then**
12. $\quad\quad (\mathsf{result}, \text{“}a = b\text{”})$
13. $\quad$ **else if** visible seq. $= (?,\clubsuit, \ldots, \overbrace{\clubsuit,\heartsuit}^{r\text{-th}}, \overbrace{\clubsuit,\clubsuit}^{(r+1)\text{-st}}, \ldots, ?,\clubsuit)$ **then**
14. $\quad\quad (\mathsf{result}, \text{“}a > b\text{”})$

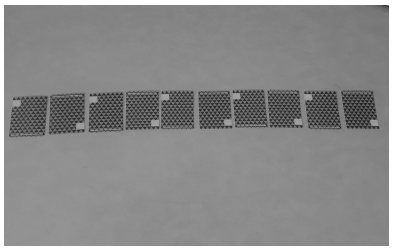
2.3. A Millionaire Protocol Using a Standard Deck

Remember that our implementation with RC explained in Section 2.2 uses a deck of black and red cards. As mentioned before, it would be great if we can perform the same task using a standard deck of playing cards instead of using a two-colored deck.

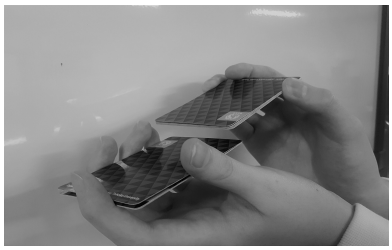
In this section, we first show that simply replacing the two-colored deck with a standard deck of cards does not work. That is, such a straight-forward protocol leaks information about Bob's input, as shown in Section 2.3.1. In Section 2.3.2, we construct a subprotocol as a new technique that prevents Alice from knowing Bob's input b. Based on this, we present the full description of our protocol using a standard deck of playing cards in Section 2.3.3.

2.3.1. Toward Using a Standard Deck of Cards

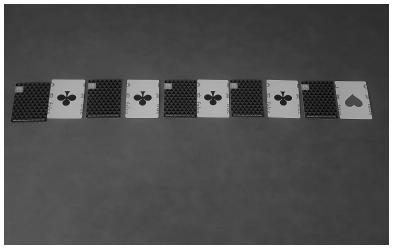
Let us first define a total order that captures a standard deck of playing cards. A *standard deck of playing cards* is a 52-card deck consisting of numbered cards $\boxed{1}\boxed{2}\cdots\boxed{52}$ such that no two cards have the same number.



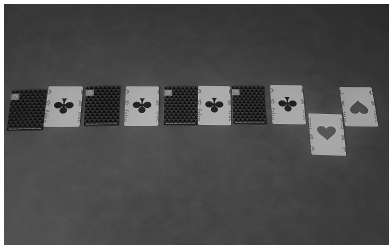
(a) Arrange Alice's input and Bob's input; the upside-down cards are put by Bob.



(b) Apply a random cut to the sequence.



(c) Bob's sequence (of upside-down cards) is revealed and then one red card appears.



(d) The left card of the red card is revealed; we have $a < b$ because the revealed card is red.

Figure 2.1: An implementation of our PP-free protocol with RC when $m = 4$

The back sides are all identical $\boxed{?}$. For convenience, in the sequel, we often regard an odd-number card as a black card $\boxed{\clubsuit}$ and an even-number card as a red card $\boxed{\heartsuit}$.

Let us execute our implementation with RC using the above standard deck instead of a two-colored deck of cards $\boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}\ \boxed{\heartsuit}\boxed{\heartsuit}\cdots\boxed{\heartsuit}$. Assume that $m = 4$. Because Alice requires four black cards and three red cards to represent her input, she has $\boxed{1}\boxed{3}\boxed{5}\boxed{7}$ and $\boxed{2}\boxed{4}\boxed{6}$. Bob has $\boxed{9}\boxed{11}\boxed{13}$ and $\boxed{8}$ as three black cards and one red card. Remembering Step 2 in which the $(m+1)$-st additional pair of cards is put, we let Alice hold $\boxed{10}$ and Bob hold $\boxed{15}$. Consider, for instance, the case where $a = 4$ and $b = 2$. Alice and Bob place the following sequences of cards:

$$\underset{\substack{5\ 3\ 7\ 1\ 10\\ \uparrow\\ a\text{-th}}}{\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}},\qquad \underset{\substack{15\ 8\ 9\ 11\ 13\\ \uparrow\\ b\text{-th}}}{\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}},$$

where Alice's black cards $\boxed{5}\boxed{3}\boxed{7}\boxed{1}$ and red card $\boxed{10}$ are randomly chosen

and placed, and Bob's sequence is arranged by shifting $\boxed{8}\boxed{9}\boxed{11}\boxed{13}\boxed{15}$. Note that Alice can memorize the order of numbers in her input. Thus, after Steps 1 and 2, we have

$$\underset{5}{\boxed{?}}\underset{15}{\boxed{\text{ }}}\underset{3}{\boxed{?}}\underset{8}{\boxed{\text{ }}}\underset{7}{\boxed{?}}\underset{9}{\boxed{\text{ }}}\underset{1}{\boxed{?}}\underset{11}{\boxed{\text{ }}}\underset{10}{\boxed{?}}\underset{13}{\boxed{\text{ }}}.$$

They apply a random cut in Step 3 and then reveal Bob's cards in Step 4. For instance, assume that the resulting sequence in Step 4 becomes the following sequence. (We here fix Bob's upside down cards for convenience.)

$$\underset{1}{\boxed{?}}\underset{10}{\boxed{?}}\underset{5}{\boxed{?}}\underset{3}{\boxed{?}}\underset{7}{\boxed{?}}$$
$$\boxed{11}\boxed{13}\boxed{15}\boxed{8}\boxed{9}.$$

Remember that $\boxed{8}$ corresponds to the red card. In this case, the position of $\boxed{8}$ placed by Bob in Step 1 is hidden due to the random cut. Therefore, replacing Bob's sequence of cards with standard cards does not cause a problem. However, a security issue occurs when Alice's card is revealed. In this example, they reveal the fourth card of Alice's input:

$$\underset{1}{\boxed{?}}\underset{10}{\boxed{?}}\underset{5}{\boxed{?}}\boxed{3}\underset{7}{\boxed{?}}$$
$$\boxed{11}\boxed{13}\boxed{15}\boxed{8}\boxed{9}.$$

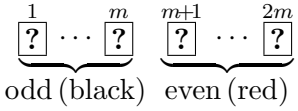
At this time, Alice gets to know that $b = 2$. That is, because Alice remembers that she placed $\boxed{3}$ at the second position in Step 1, she learns that Bob put the red card $\boxed{8}$ at the second position, which implies that $b = 2$. Therefore, if we execute our protocol presented in Section 2.2 using a standard deck of cards, the value of b would be leaked when Alice's card is revealed in Step 4. That is, this straight-forward implementation is not secure, and we need a new technique.

2.3.2. Subprotocol

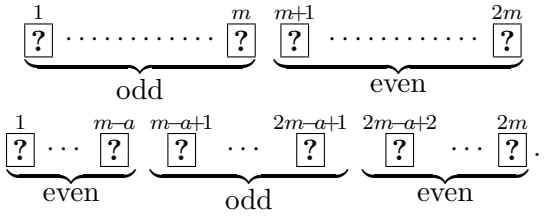
As seen in Section 2.3.1, simply replacing the deck with a standard deck of playing cards does not work (namely, the value of b would be leaked to Alice). Let us confirm again the reason why the simple replacement described in Section 2.3.1 makes Alice know Bob's input b. That occurs when Alice's b-th card is opened in Step 4 (because Alice memorizes the order of her cards). Therefore, we construct a subprotocol that hides the order of Alice's cards from her while guaranteeing the "format" of Alice's input, i.e., placing odd-number cards at every position from the first to the a-th and even-number cards from the $(a + 1)$-st to the last. If we execute such a subprotocol in advance, b is hidden from Alice even if they reveal the b-th card in Alice's input sequence because Alice does not know where the revealed card was in the input.

The subprotocol proceeds as follows:

1. Alice and Bob shuffle m odd-number cards (corresponding to black) and then place them on the table with their faces down. Similarly, they shuffle m even-number cards (corresponding to red) and then place them:

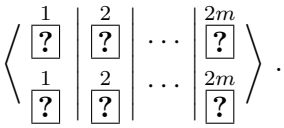


2. Alice takes $m+1$ odd-number cards and $m-1$ even-number cards from the deck. Then, she places a sequence of cards consisting of odd-number cards at positions from the $(m-a+1)$-st to the $(2m-a+1)$-st (in any order) and even-number cards at the remaining positions (in any order) below the sequence of cards placed in Step 1 with their faces down:



   Note that the subsequence just above the $(m+1)$ odd-number cards placed now becomes an encode of the value of a. We call the sequence placed by Alice the sequence in the second row.

3. Considering the two cards in the same column as a pile, apply a *pile-shifting shuffle* to the sequence of piles:

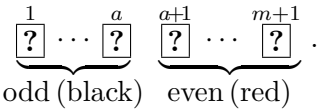


   This cyclically shuffles a sequence of piles. To implement this shuffle, we use a physical case that can store a pile of cards, such as boxes and envelopes [61]; one cyclically shuffles them by hand until nobody can trace the offset.[iv]

4. Reveal the sequence in the second row. The subsequence of $m+1$ cards above the revealed odd-number cards represents a. The revealed $2m$ cards can be reused.

[iv] This operation is the same as Bob puts his cards upside down and then they apply a random cut (as described in Section 2.2.1). If the back-side symbol of the standard deck of cards is vertically asymmetric, using a random cut is more efficient because an additional tool is not required.

Thus, we can obtain a sequence of $m+1$ cards representing $a \in \{1, 2, \ldots, m\}$ by executing the above subprotocol:



Note that, by virtue of the shuffle in Step 1, neither Alice nor Bob can know the order of the obtained sequence of the number cards encoding Alice's value a in Step 4. Because they apply a pile-shifting shuffle in Step 3, revealing the sequence in the second row does not leak any information about the value of a. More specifically, we show that no information about Alice's input a is leaked during the execution of the subprotocol, as follows. In Steps 1, 2, and 3, a is not leaked because players just place cards with their faces down and publicly apply shuffle operations. In Step 4, the sequence of cards in the second row placed in Step 2 is revealed. Then, odd-number cards appear from the $(m-a+1+r)$-th to the $(2m-a+1+r)$-th (apart from cyclic rotation) where $r \in \{0, 1, \ldots, 2m-1\}$ is a random value generated by the pile-shifting shuffle in Step 3. Therefore, a is not leaked by revealing the sequence of cards in the second row.

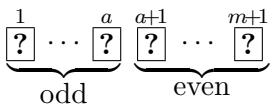
The total number of shuffles required for the subprotocol is three because two shuffles in Step 1 and one shuffle in Step 3 are applied. The number of required cards is $4m$.

2.3.3. Description

We are now ready to present our protocol using a standard deck of playing cards; our implementation is obtained by combining the subprotocol explained in Section 2.3.2 with the PP-free protocol with RC explained in Section 2.2.

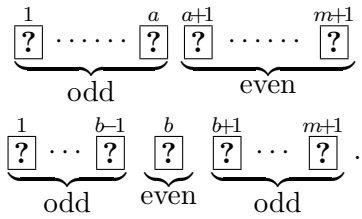
The protocol proceeds as follows:

1. Execute the subprotocol explained in Section 2.3.2 to obtain Alice's input sequence of cards:

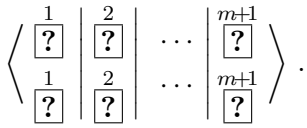


   The revealed $2m$ cards in the subprotocol can be reused; Bob takes them and then places $m+1$ cards (namely, Bob's input sequence) below Alice's, such that only the b-th card has an even number and the remaining m odd-number cards starting at the $(b+1)$-st position (apart from cyclic rotation) are sorted in ascending order (or in any

order):



2. Apply a pile-shifting shuffle:



3. Reveal Bob's cards. Then, one even-number card appears. Reveal the card just above the even-number card.
   - We have $a < b$ if the revealed card is even.
   - We have $a \geq b$ if the revealed card is odd. To see whether equality holds or not, open the card to the right of the revealed card. If the opened card is even, we have $a = b$. If it is odd, we have $a < b$.

Thus, we can solve the millionaire problem by executing the above protocol. In Steps 1 and 2, information about Alice's input a and Bob's input b is not leaked because they just execute the subprotocol, which is shown to be secure in Section 2.3.2, and publicly apply a shuffle operation. In Step 3, Bob's input sequence is revealed at first. Then, one even-number card appears in the $(b + r)$-th position (apart from cyclic rotation) where $r \in \{0, 1, \ldots, m\}$ is a random value generated by the pile-shifting shuffle in Step 2. Therefore, the value of b is not leaked by revealing Bob's input sequence. Then, the card just above the revealed even-number card, i.e., the b-th card in Alice's input sequence is revealed. This revealed card does not leak b because nobody knows where the revealed card was placed in the input due to the subprotocol. Therefore, the value of b is not leaked by revealing the b-th card in Alice's input sequence. Opening the $(b + 1)$-st card of Alice's input sequence is similar.

2.3.4. The Efficiency

The total number of shuffles required for our implementation using a standard deck is four because three shuffles in the subprotocol and one shuffle in Step 2 are applied. The number of required cards is $4m$ because they

reuse cards to execute the protocol after executing the subprotocol with $4m$ cards.[v] See Table 2.1 again.

2.4. The Existing Circuit-Based Protocol

Hereinafter, we deal with another approach for solving the millionaires' problem: We introduce the existing circuit-based protocol [52] in this section, and then we will improve upon it in the next section.

Consider the following encoding:

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0,\ \boxed{\heartsuit}\boxed{\clubsuit} = 1. \tag{2.1}$$

Then, Alice and Bob can place sequences of cards corresponding to the binary representations of $a=(a_n,\ldots,a_1)_2$ and $b=(b_n,\ldots,b_1)_2$, respectively, where $n=\lceil \log_2 m \rceil$:

$$\underbrace{\boxed{?}\boxed{?}}_{a_n}\cdots\underbrace{\boxed{?}\boxed{?}}_{a_1}\ \ \underbrace{\boxed{?}\boxed{?}}_{b_n}\cdots\underbrace{\boxed{?}\boxed{?}}_{b_1}.$$

Such a pair of face-down cards

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

corresponding to a bit $x \in \{0,1\}$ is called a *commitment* to x. Given the above card sequence along with some additional cards, the existing circuit-based protocol given by Nakai et al. [52] determines whether $a < b$ or not:

$$\underbrace{\boxed{?}\boxed{?}}_{a_n}\cdots\underbrace{\boxed{?}\boxed{?}}_{a_1}\underbrace{\boxed{?}\boxed{?}}_{b_n}\cdots\underbrace{\boxed{?}\boxed{?}}_{b_1}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\to\cdots\to\underbrace{\boxed{?}\boxed{?}}_{\mathrm{bool}(a<b)},$$

where $\mathrm{bool}\,(a<b)$ represents

$$\mathrm{bool}\,(a<b) \coloneqq \begin{cases} 0 & \text{if } \ a \geq b, \\ 1 & \text{if } \ a < b. \end{cases}$$

Their protocol proceeds based on the logical circuit shown in Protocol 2.

To implement that circuit, one requires AND (OR) and COPY protocols; Nakai et al. [52] used the six-card AND protocol [48], producing a commitment to $x \wedge y$ from the input commitments to x and y:

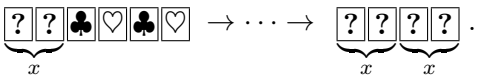
$$\underbrace{\boxed{?}\boxed{?}}_{x}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{y}\ \to\cdots\to\ \underbrace{\boxed{?}\boxed{?}}_{x\wedge y},$$

[v] Note that the computational model of card-based protocols [45] assumes that all inputs are given at the beginning. If we follow this assumption, i.e., Bob should put input cards at the beginning of the protocol, the number of required cards is $5m$.

Protocol 2. *The circuit-based protocol* [52]

input: $a = (a_n, \ldots, a_1)_2, b = (b_n, \ldots, b_1)_2$;
$f_1 = \bar{a}_1 \wedge b_1$;
for $(i : 2$ to $n)$ {
$f_i = (\bar{a}_i \wedge b_i) \vee ((\bar{a}_i \vee b_i) \wedge f_{i-1})$; }
output : f_n $(= \mathrm{bool}\,(a < b))$.

and the six-card COPY protocol [48], producing two commitments to x from an input commitment to x:



Let us count the number of required cards for implementing the circuit. First, two additional cards ♣♡ are required to compute $f_1 = \bar{a_1} \wedge b_1$ using the AND protocol [48]. Note that we have four reusable cards ♣♣♡♡ directly after computing f_1. Then, because six additional cards ♣♣♣♡♡♡ are required to duplicate the commitments to a_2 and b_2 in order to compute $f_2 = (\bar{a}_2 \wedge b_2) \vee ((\bar{a}_2 \vee b_2) \wedge f_1)$, another two cards ♣♡ are required. Consequently, four additional cards ♣♣♡♡ are necessary before computing f_1, and hence, the total number of required cards is $4\lceil \log m \rceil + 4$. (Note that directly after computing f_j, $2 \leq j \leq n-1$, we have enough reusable cards to compute f_{j+1}, i.e., four additional cards are sufficient to implement the circuit.)

Next, let us count the number of required shuffles. Note that each of the AND [48] and COPY [48] protocols uses one shuffle and we have two reusable cards after the protocol terminates; furthermore, we can obtain two more reusable cards after using the AND protocol [48] if we apply one more shuffle (as seen in Appendices A and B). First, one shuffle is required to compute f_1, and one more shuffle is required to produce four reusable cards. Then, the circuit-based protocol uses the COPY protocol [48] twice and the AND (OR) protocol [48] four times to compute f_i, $2 \leq i \leq n$. Moreover, one more shuffle is required to produce reusable cards enough to compute f_{i+1}. That is, seven shuffles are required to compute f_j, $2 \leq j \leq n-1$, and six shuffles are required to compute f_n. Consequently, the total number of required shuffles is $7\lceil \log m \rceil - 6$.

It should be noted that, as seen above, this existing circuit-based protocol produces a commitment to $\mathrm{bool}\,(a < b)$ (while our implementations with RC presented in Sections 2.2 and 2.3 reveal the value of it to the players at the end of the protocols). Therefore, one can reuse the commitment in a larger protocol.

2.5. Our Improved Circuit-Based Protocol

In this section, we improve upon the circuit-based protocol introduced in Section 2.4, i.e., we present an improved circuit-based protocol that uses a less number of shuffles and cards. We first show the idea behind our improved circuit-based protocol in Section 2.5.1, and then show the procedure of our improved circuit-based protocol in Section 2.5.2.

2.5.1. Idea

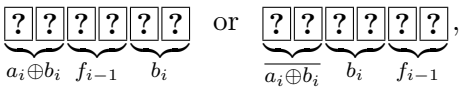
We borrow the idea behind the storage protocol [52]; it uses Private Permutation and regards f_i shown in Section 2.4 as:

$$f_i = \begin{cases} f_{i-1} & \text{if } a_i = b_i, \\ b_i & \text{if } a_i \neq b_i. \end{cases} \tag{2.2}$$

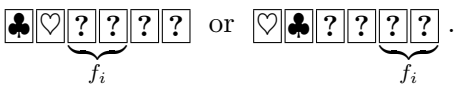
That is, the storage protocol is supposed to choose f_{i-1} or b_i depending on whether $a_i = b_i$ or not. More specifically,

- f_{i-1} is equal to bool $((a_{i-1}, \ldots, a_1) < (b_{i-1}, \ldots, b_1))$;
- $a_i = b_i$ implies $f_i = f_{i-1}$;
- $a_i = 0$ and $b_i = 1$ imply $(a_i, \ldots, a_1) < (b_i, \ldots, b_1)$, and hence $f_i = 1 = b_i$ while $a_i = 1$ and $b_i = 0$ imply $(a_i, \ldots, a_1) > (b_i, \ldots, b_1)$, and hence $f_i = 0 = b_i$.

Such a choice can be made without Private Permutation; if we can let a six-card sequence be either

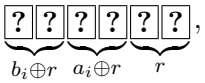


then, we can obtain a commitment to f_i by revealing the first two cards as follows:

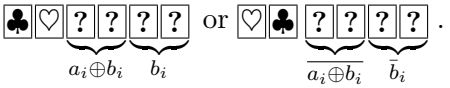


The above flow can be accomplished by using the procedure of the six-card AND protocol [48].

Moreover, we can easily obtain commitments to $a_i \oplus b_i$ and b_i by using the six-card COPY protocol [48]. That is, from the following sequence where r is a uniform random bit:



it is determined whether $r = b_i$ or $r = \bar{b}_i$ by revealing the first two cards, and then we obtain commitments to $a_i \oplus b_i$ and b_i as:

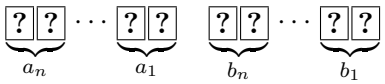


Note that revealing the first two cards leaks no information about b_i because r is a random bit.

As described above, by using the procedure of the COPY and AND protocols [48], we can obtain a commitment to f_i according to Eq. (2.2) without revealing the values of a_i, b_i, and f_{i-1}. It should be noted that f_i in Eq. (2.2) is the three-input majority function of a_i, b_i, and f_{i-1} . An efficient card-based protocol for the three-input majority function was proposed by Nishida et al. [60] in 2013, which was based on the same idea mentioned above.

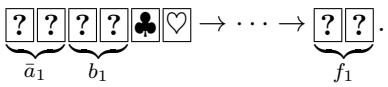
2.5.2. The Description of Our Protocol

Based on the idea presented in Section 2.5.1, we construct an improved circuit-based protocol. Given the input card sequence



and two additional cards, our protocol proceeds as follows.

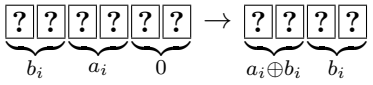
1. Compute $f_1 = \bar{a}_1 \wedge b_1$ by using the six-card AND protocol [48]:



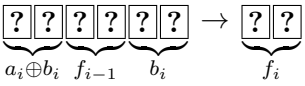
   Now, two reusable cards remain.

2. Repeat the following computation from $i = 2$ to $i = n$.

   (a) Obtain commitments to $a_i \oplus b_i$ and b_i from the commitments to a_i and b_i by using the six-card COPY protocol [48] (and the NOT computation):



   (b) Obtain a commitment to f_i from the commitments to $a_i \oplus b_i$, b_i, and f_{i-1} by using the six-card AND protocol:



3. Then, a commitment to $f_n = \mathrm{bool}\,(a < b)$ can be obtained:

$$\underbrace{\boxed{?}\boxed{?}}_{\mathrm{bool}(a<b)} \ .$$

Due to the use of two additional cards, this protocol uses $(4\lceil \log m \rceil + 2)$ cards in total. The number of required shuffles is $2\lceil \log m \rceil - 1$ in total, because this protocol repeats each procedure of the AND protocol and the COPY protocol from $i = 2$ to $i = n$ after AND computation for f_1, as shown in Table 2.1.

This improved circuit-based protocol is a combination of the existing information theoretically secure card-based protocols, and hence, it is guaranteed to be secure.

2.6. Circuit-based Protocols with a Standard Deck

In this section, let us run the circuit-based protocols presented in Sections 2.4 and 2.5 with a standard deck of cards. To achieve this, we consider the use of the existing elementary protocols with a standard deck of cards, namely the AND and COPY protocols listed in Table 2.2.

First, combine these elementary protocols with the (NTMIO) circuit-based protocol [52] presented in Section 2.5. Table 2.3 shows the numbers of required cards and shuffles; they can be counted in a similar way to Section 2.4.

Next, let us run our improved circuit-based protocol explained in Section 2.5 with a standard deck of cards. Because Step 2(b) of our protocol uses not a normal AND computation but a special choice of cards, we cannot use the elementary AND protocols straightforwardly. Fortunately, we found that the choice can be made by the (extended) existing AND protocol [39] easily. The extended protocol requires four additional cards and five shuffles.[vi] Therefore, the resulting circuit-based protocol uses $(4\lceil \log m \rceil + 4)$ cards and $(6\lceil \log m \rceil - 2)$ shuffles in total.

2.7. Conclusion

In this chapter, we proposed three card-based protocols to solve the millionaires' problem without using Private Permutation. See Table 2.1 again for the performance of the PP-free millionaire protocols. In particular, the PP-free protocol with RC proposed in Section 2.2 uses only one random cut, and its correctness and secrecy are clear. Therefore, we believe that even

[vi] More specifically, the extended protocol proceeds in a similar way to the existing AND protocol [39] except for producing "opaque" commitments to f_{i-1} and b_i; refer to the paper [39] for details.

Table 2.2: The existing elementary protocols with a standard deck of cards. Note that the number of required shuffles for each of the AND protocol proposed by Niemi and Renvall [57] and the one proposed by Koch, Schrempp, and Kirsten [29] is an expected value.

	Function	**#Cards**	**#Shuffles**
Niemi–Renvall [57]	AND	5	9.5
Mizuki [39]	AND	8	4
Koch–Schrempp–Kirsten [29]	AND	4	6
Mizuki [39]	COPY	6	1

Table 2.3: The NTMIO circuit-based protocol with a standard deck of cards using the existing elementary protocols [29, 39, 57]

	#Cards	**#Shuffles**
NTMIO with Niemi–Renvall [57]	$4\lceil \log m \rceil + 4$	$40\lceil \log m \rceil - 30.5$
NTMIO with Mizuki [39]	$4\lceil \log m \rceil + 6$	$18\lceil \log m \rceil - 14$
NTMIO with Koch–Schrempp–Kirsten [29]	$4\lceil \log m \rceil + 4$	$26\lceil \log m \rceil - 20$

non-experts such as high school students can easily understand and use it practically. Note that a random cut can be easily and securely implemented by using the Hindu cut [81]. When preparing a two-colored deck of cards is hard, our protocol presented in Section 2.3 will be benefical because it can be executed with a standard deck of cards, which is sold almost all over the world.

Moreover, we can use our protocols in didactic contexts in order to invite young people and students to cryptography; they would be an ideal tool to exhibit the concept of secure multiparty computations, as often pointed out, e.g., [20, 36].

Regarding the number of required cards, one might think of the use of the existing four-card AND [31], the five-card AND [2], and the five-card COPY protocols [62] in the previous circuit-based protocol because the number of required cards should be reduced. However, the numbers of required shuffles for the four-card and five-card AND protocols are eight and seven on average, respectively. Moreover, the five-card COPY protocol requires ideal cases for execution. Therefore, for practicality, we considered only the six-card AND protocol [48] and the six-card COPY protocol [48].

3. ZKP for Puzzles

Sudoku is one of the most famous puzzles. In a standard challenge, a 9×9 grid is used, which is divided into 3×3 subgrids. Some of the cells are already filled with numbers between 1 and 9. The goal of Sudoku is to fill all the empty cells with numbers so that each row, each column, and each subgrid contains all the numbers from 1 to 9. Figure 3.1 shows an example of a standard Sudoku challenge, and its solution.

We address a generalized version of Sudoku in this study. That is, a Sudoku puzzle where a grid is $n \times n$ cells, a subgrid is $k \times k$ cells, and numbers from 1 to n are used. Note that $n = k^2$; the standard size of a Sudoku puzzle corresponds to $n = 9$ and $k = 3$.

We solicit zero-knowledge proof protocols for Sudoku. That is, for a certain Sudoku puzzle, we assume a prover P who knows the solution to the Sudoku puzzle and a verifier V who does not know it, and suppose that P wants to convince V of the following without revealing any information about the solution:

- There is a solution to the puzzle;
- P knows the solution.

Unlike in conventional cryptographic zero-knowledge proofs (see, e.g., [18]), in our setting, we do not want to use electronic devices such as computers and network devices. Instead, we want to use only everyday items to execute a protocol manually. Further, the prover P and the verifier V are assumed to be in the same place. Such a restricted zero-knowledge proof is called a *physical zero-knowledge proof* [5, 8, 19].

In 2009, Gradwohl, Naor, Pinkas, and Rothblum proposed several physical zero-knowledge proof protocols for Sudoku [19]. Among them, Protocol 3

		1			5	6	7	
	2				4	8		
6	7							
3				5				
				4			1	8
					8	2		9
					2	4		
	9	2			7		8	3
	6		1					2

8	3	1	9	2	5	6	7	4
9	2	5	6	7	4	8	3	1
6	7	4	8	3	1	9	2	5
3	1	8	2	5	9	7	4	6
2	5	9	7	4	6	3	1	8
7	4	6	3	1	8	2	5	9
1	8	3	5	9	2	4	6	7
5	9	2	4	6	7	1	8	3
4	6	7	1	8	3	5	9	2

Figure 3.1: Example of the standard Sudoku challenge, and its solution

(hereinafter referred to as *GNPR Protocol 3*) utilizes a deck of cards having numbers on their faces, such as playing cards[i]. This protocol needs $3n^2$ cards, and it has an extractability error, i.e., a prover P who does not know the solution can convince V with a non-zero probability. By contrast, Protocol 5 (hereinafter referred to as *GNPR Protocol 5*) avoids extractability error by utilizing special cards (namely, scratch-off cards that allow the colors to be covered) together with scissors[ii]. Unfortunately, GNPR Protocol 5 consumes non-reusable scratch-off cards at every execution of the protocol. Therefore, it is preferable to construct a protocol that can be implemented with only reusable everyday objects such as playing cards.

Contribution

In this chapter, we propose new zero-knowledge proof protocols that satisfy the following: (i) they utilize the same items as GNPR Protocol 3, namely a standard deck of playing cards, and (ii) they have no extractability error, implying that if P does not know a solution, P cannot convince V. The main idea behind our protocols is to apply techniques of card-based cryptography (e.g., [22,23,46]). Table 3.1 shows a comparison of performances of GNPR Protocol 3 and our three protocols, which we call Protocols A, B, and C; recall that we assume an $n \times n$ Sudoku grid. As known from Table 3.1, we will design three protocols considering a tradeoff between the number of cards and the number of shuffles. The most important point is that our protocols have no extractability error, i.e., they are "perfectly extractable." As will be explained in Section 3.1.1, for physical zero-knowledge protocols, perfect extractability is crucial in terms of efficiency and feasibility.

Moreover, we propose a physical ZKP protocol for Slitherlink, which is not like other Nikoli's puzzles since it requires to draw a **single** loop to solve the puzzle. This feature of the puzzle is a challenge that was not present in the previous physical ZKPs for Nikoli's puzzles [5,6,8,14,19,72].

Outline

The remainder of this chapter is organized as follows. In Section 3.1, we review zero-knowledge proof of knowledge and GNPR Protocol 3. In Section 3.2, we present our proposed protocols for Sudoku, named Protocols A and B, which are designed with the help of "copy technique." In Section 3.3, we introduce another idea of utilizing interaction so as to have Protocol C. In Section 3.4, we change the target puzzle to Slitherlink. In Section 3.5 we present our ZKP protocol for Slitherlink. In Section 3.6, we prove the

[i] Protocols 1 and 2 presented in [19] are conventional (i.e., non-physical) zero-knowledge proof protocols.

[ii] Protocol 4 in [19] is a variation of GNPR Protocol 3.

Table 3.1: Comparison of protocols

	#Cards	#Shuffles	Extractability Error
GNPR Protocol 3	$3n^2$	$3n$	at most 1/9
Our Protocol A	$n^2 + n$	$5n$	0
Our Protocol B	$2n^2 + n$	$4n$	0
Our Protocol C	$3n^2$	$3n + 1$	0

three properties of our proposed protocol. In Section 3.7, we conclude this chapter.

3.1. Preliminaries

In this section, we first review zero-knowledge proof and then introduce the existing protocol, GNPR Protocol 3.

3.1.1. Zero-Knowledge Proof

A zero-knowledge proof is an interactive proof between a prover P and a verifier V. Formally, they both have an instance $\mathcal{I}$ of a problem and only P knows a solution w. In addition, V is computationally bounded so that V cannot obtain w from $\mathcal{I}$. Under these assumptions, P wants to convince V that he/she knows w without revealing any information about w. Such a proof is called a *zero-knowledge proof*, which must satisfy the following three properties.

Completeness If P knows w, P can convince V.

Extractability If P does not know w, P cannot convince V (with a high probability).

Zero-knowledge V cannot obtain any information about w. In order to prove that a protocol satisfies this property, assuming a probabilistic polynomial time algorithm $M(\mathcal{I})$ not having w, if outputs of the protocol and outputs of $M(\mathcal{I})$ follow the same probability distribution, the zero-knowledge property is satisfied.

The probability that V will be convinced although P does not know w is called the *extractability error*. If we have a zero-knowledge proof protocol, the extractability error of which is $\delta > 0$, repeating the protocol ℓ times allows V to detect that P does not know w with a probability $1 - \delta^\ell$. Therefore, in general, even if a protocol has a non-zero extractability error, i.e., it

is not *perfectly extractable*, we can in practice establish zero-knowledge proof with a negligible extractability error by repeating the protocol. However, since we assume that a protocol is executed by human hands, it is impractical to repeat the protocol many times. Therefore, it is indispensable to design a protocol having no extractability error.

A zero-knowledge proof was first defined by Goldwasser, Micali, and Rackoff [18], and it was proved that (computational) zero-knowledge proofs exist for any NP problems [17]. Because it is known that Sudoku is NP-complete [84], we can construct conventional (computational) zero-knowledge proof protocols for it [19]. Remember, however, that this paper focuses not on conventional zero-knowledge proof but on physical zero-knowledge proof for Sudoku. Hence, we introduce the existing physical protocol, GNPR Protocols 3, in the next subsection.

3.1.2. Gradwohl, Naor, Pinkas, and Rothblum Protocol 3

Here, we review GNPR Protocol 3 [19]. This protocol utilizes physical cards, the face side of each of which has one number between 1 and n, such as $\boxed{1}\,\boxed{2}\,...\,\boxed{n}$; all the back sides are identical, $\boxed{?}\,\boxed{?}\,...\,\boxed{?}$. The protocol uses $3n$ sets of such n cards, namely, $3n^2$ cards in total.

Before presenting the protocol, we define a shuffle operation for cards. Given a sequence of m face-down cards $(c_1, c_2, c_3, ..., c_m)$, a *shuffle* results in a sequence

$$\left(c_{r^{-1}(1)}, c_{r^{-1}(2)}, c_{r^{-1}(3)}, ..., c_{r^{-1}(m)}\right),$$

where $r \in S_m$ is a uniformly distributed random permutation, and S_m is the symmetric group of degree m.

GNPR Protocol 3 proceeds as follows.

1. The prover P places three face-down cards on each cell according to the Sudoku solution. On the filled-in cells, P places three face-up cards corresponding to the numbers filled in. After V confirms the values of the face-up cards, P turns them over.

2. The verifier V picks one card randomly from each cell of a row to make a packet of n cards corresponding to the row. Because there are n rows, n packets are created. The same procedure is applied for each column and each subgrid. Thus, V makes $3n$ packets in total and passes them to P.

3. P who received the packets from V applies a shuffle to the cards in each packet and returns the $3n$ shuffled packets to V.

4. V opens all the cards in all the packets and checks that each packet contains all the numbers from 1 to n.

This is GNPR Protocol 3, which satisfies the three properties of zero-knowledge proof, as follows.

Completeness: If P places the face-down cards correctly according to the solution, every packet made by V must contain all the numbers from 1 to n. By checking them, V is convinced that all the cards have been placed according to the solution. Furthermore, V is convinced that the packets are not a solution to another puzzle instance, because V saw the face-up cards placed on filled-in cells be compatible with the given puzzle instance.

Extractability: Consider a situation where V is convinced in spite of an illegal input by P. Such a situation occurs when the three cards placed on each cell are not identical; it was proved that the probability that V is convinced in this case is at most $1/9$ [19]. Thus, the GNPR Protocol 3 has an extractability error, i.e., it is not perfectly extractable.

Zero-knowledge: If all distribution of revealed cards during an execution of a protocol can be simulated by a simulator $M(\mathcal{I})$, the protocol satisfies zero-knowledge. In GNPR Protocol 3, each packet is opened in Step 4. Since a shuffle operation has been applied for every packet in Step 3, the order of each sequence is uniformly distributed on S_n. Therefore, $M(\mathcal{I})$ can simulate the outputs.

In this GNPR Protocol 3, P places three cards on each cell, and hence, the protocol uses $3n^2$ cards in total. For example, in the case of a Sudoku puzzle consisting of a 9×9 grid, this protocol needs 243 cards. Because this protocol has an extractability error, we have to repeat the protocol many times to make the extractability error negligible. As mentioned in Section 3.1, a protocol which has no extractability error is preferable. In the next section, we propose such two perfectly extractable protocols with fewer cards.

3.2. Proposed Protocols A and B for Sudoku: Applying Copy Technique

In this section, we propose two efficient zero-knowledge proof protocols for Sudoku with no extractability error in which card-based cryptography perspectives are applied. For easy to explain our protocols, we first define some terms in Section 3.2.1 and some subprotocols in Sections 3.2.2 and 3.2.3. Then, we construct our first protocol called Protocol A in Section 3.2.4 and our second protocol called Protocol B in Section 3.2.5. The outline of both protocols is as follows:

1. P places exactly one face-down card on each cell corresponding to the solution;
2. V checks that the format of the packet of face-down cards placed on each row, each column and each subgrid is correct.

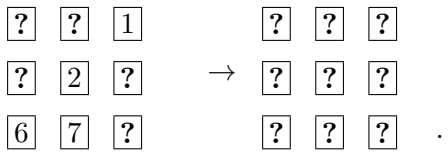
In Section 3.2.6, we show that our protocols satisfy the zero-knowledge proof properties.

3.2.1. Terminologies

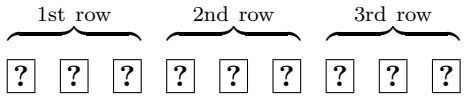
In this subsection, we present some terms.

Commitment as Input. In our protocols, the prover P places a single face-down card on each cell according to the solution. On filled-in cells, P places face-up cards. After V confirms the value of the face-up cards, P turns them over. Now, there are exactly n^2 cards placed on the grid. We call a sequence of n face-down cards corresponding to each row, each column or each subgrid a *commitment*.

For example, in the case of the top-left subgrid in Figure 3.1, P places nine cards according to the solution; after V confirms the value of the face-up cards, P turns them over:



This is a commitment corresponding to this subgrid, and we regard it as a sequence:



Pile-scramble Shuffle We here introduce a well-known shuffle operation called *pile-scramble shuffle* [24]. Given m piles, each of which consists of the same number of face-down cards, we denote this by

$$(pile_1, pile_2, pile_3, \ldots, pile_m)\,.$$

For such a sequence of piles, applying a pile-scramble shuffle results in

$$\left(pile_{r^{-1}(1)}, pile_{r^{-1}(1)}, pile_{r^{-1}(3)}, \ldots, pile_{r^{-1}(m)}\right),$$

where $r \in S_m$ is a uniformly distributed random permutation. A pile-scramble shuffle can be implemented with the help of clips, envelopes, or

similar items. More specifically, for example, we put each pile of cards in an envelope and then shuffle the envelopes until nobody can trace the move; if players have difficulty in shuffling the envelopes, they may put all the envelops in a large box, close the box, and randomly rotate it to scramble the envelopes inside.

3.2.2. Verification Protocol

In this subsection, we construct a protocol which verifies that a given commitment contains all the numbers without revealing its order, and furthermore, produces the same commitment as the original one by using additional n cards. To construct this protocol, we borrow two existing ideas: (i) a method for regarding a commitment as a permutation and a technique for inverting a permutation, which were given partially by Hashimoto, Shinagawa, Nuida, Imamura, and Hanaoka [22] and Ibaraki and Manabe [23], and (ii) a technique for checking the format of a sequence of face-down cards, which was given by Mizuki and Shizuya [46]. That is, we regard a commitment consisting of n cards as a permutation $v \in S_n$:

$$\boxed{?}\ \boxed{?}\ \boxed{?}\ \cdots\ \boxed{?}\quad (v),$$

where a card having number i, $1 \leq i \leq n$, on its face side is placed at the $v(i)$-th position, and a permutation with parentheses, such as (v), means that the permutation is hidden (because the cards are face-down).

Given a commitment corresponding to $v \in S_n$, our *verification protocol* proceeds as follows.

1. V puts n cards numbered from 1 to n in this order to generate a card sequence corresponding to the identity permutation id under the given commitment to v:

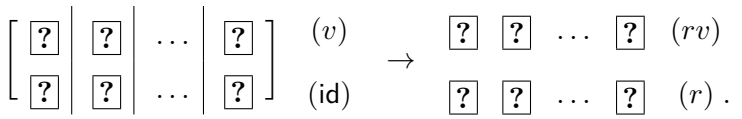
$$\begin{array}{llllll} \boxed{?} & \boxed{?} & \boxed{?} & \cdots & \boxed{?} & (v) \\ \boxed{1} & \boxed{2} & \boxed{3} & \cdots & \boxed{n} & \mathsf{id}\ . \end{array}$$

2. V turns over all the face-up cards in the bottom row and stacks the cards in each column so that there are n two-card piles:

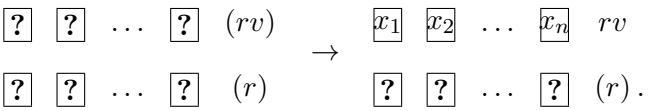
$$\begin{array}{c|c|c|c|l} \boxed{?} & \boxed{?} & \cdots & \boxed{?} & (v) \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} & (\mathsf{id})\ . \end{array}$$

3. P applies a pile-scramble shuffle to them and obtains a commitment to $rv \in S_n$ and a commitment to $r \in S_n$, where $r \in S_n$ is a uniformly

distributed random permutation:

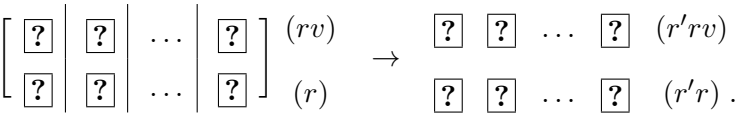


4. V reveals the commitment to rv in the top row; let $x_1, x_2, \ldots, x_n$ be the numbers written on the revealed cards in this order:

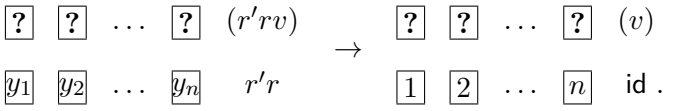


If they consist of all numbers from 1 to n, V is convinced that the original commitment was in a correct format. Further, since V learns only the value of rv, which is also a random permutation, no information about v leaks.

5. From now on, we reconstruct the original commitment to v. First, P turns over all the opened cards and applies a pile-scramble shuffle to them again. After that, a commitment to rv becomes $r'rv \in S_n$, and a commitment to r becomes $r'r \in S_n$, where $r' \in S_n$ is a uniformly distributed random permutation:



6. V reveals the bottom row; let $y_1, y_2, \ldots, y_n$ be the revealed numbers. Then, V sorts the columns so that the bottom row becomes id:



This means that permutation $(r'r)^{-1}$ is multiplied to each row, and hence, the top row becomes the same commitment as the original one to v.

This is our verification protocol, which can verify the format of a given commitment without destroying it.

3.2.3. Copy Protocol

Next, we present our *copy protocol*, which is an extended version of the verification protocol constructed in the previous subsection. Given a commitment to v, this protocol makes two identical copied commitments to v,

and furthermore, verifies that the given commitment contains all the numbers. Given a commitment to $v \in S_n$ along with $2n$ additional cards, the copy protocol proceeds as follows.

1. P and V apply the same procedure as Steps 1 to 4 in the verification protocol. After that, they obtain a card sequence rv and a commitment to r, where $r \in S_n$ is a uniformly distributed random permutation:

$$\begin{array}{cccccc} \boxed{?} & \boxed{?} & \dots & \boxed{?} & (v) & \\ \boxed{1} & \boxed{2} & \dots & \boxed{n} & \mathsf{id} & \end{array} \quad \rightarrow \quad \begin{array}{cccccc} \boxed{x_1} & \boxed{x_2} & \dots & \boxed{x_n} & rv & \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r) & . \end{array}$$

2. Using additional n cards, V generates the same card sequence as the top row and puts it at the top so that we have

$$\begin{array}{cccccc} \boxed{x_1} & \boxed{x_2} & \dots & \boxed{x_n} & rv & \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r) & \end{array} \quad \rightarrow \quad \begin{array}{cccccc} \boxed{x_1} & \boxed{x_2} & \dots & \boxed{x_n} & rv & \\ \boxed{x_1} & \boxed{x_2} & \dots & \boxed{x_n} & rv & \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r) & . \end{array}$$

3. In a similar manner to Step 5 in the verification protocol, V turns over all the opened cards and applies a pile-scramble shuffle to them. After that, for a uniformly distributed random permutation $r' \in S_n$, we have

$$\left[\begin{array}{c|c|c|c} \boxed{?} & \boxed{?} & \dots & \boxed{?} \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} \end{array}\right] \begin{array}{c} (rv) \\ (rv) \\ (r) \end{array} \quad \rightarrow \quad \begin{array}{cccccc} \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r'rv) & \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r'rv) & \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r'r) & . \end{array}$$

4. Finally, V applies a similar procedure to Step 6 in the verification protocol. More precisely, V reveals the bottom row to see the numbers $y_1, y_2, \dots, y_n$, and V sorts the columns so that the bottom row becomes id. Then, they obtain two copied commitments to v:

$$\begin{array}{cccccc} \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r'rv) & \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} & (r'rv) & \\ \boxed{y_1} & \boxed{y_2} & \dots & \boxed{y_n} & r'r & \end{array} \quad \rightarrow \quad \begin{array}{cccccc} \boxed{?} & \boxed{?} & \dots & \boxed{?} & (v) & \\ \boxed{?} & \boxed{?} & \dots & \boxed{?} & (v) & \\ \boxed{1} & \boxed{2} & \dots & \boxed{n} & \mathsf{id} & . \end{array}$$

This is the copy protocol; notice that it uses $2n$ additional cards.

3.2.4. Protocol A

We are now ready to describe our first protocol, Protocol A. It proceeds as follows.

1. The prover P places a single card on each cell according to the solution (as described in Section 3.2.1).

2. V regards the n cards placed on each row as a commitment and applies the verification protocol (described in Section 3.2.2) to the commitment. By doing so, V can verify whether each commitment contains all numbers. Note that the revealed cards in the verification protocol are reuseable.

3. For the n cards placed in each column, V does the same procedure as Step 2.

4. As in a similar way to GNPR Protocol 3, make n packets corresponding to n subgrids. Each packet is shuffled. V opens all the packets and checks that each packet includes all numbers from 1 to n.

Let us count how many cards we use in this Protocol A. After P inputs the cards in Step 1, there are n^2 cards on the grid. For applying the verification protocol, we need n additional cards, and hence, we need n^2+n cards in total. Next, let us count the number of shuffles in this protocol. Every execution of the verification protocol requires two pile-scramble shuffles, and it is performed for each of n rows and n columns. The packet shuffle in Step 4 is performed once for each subgrid, and there are n subgrids. Therefore, the total number of shuffles is $5n$. See Table 3.1 again.

3.2.5. Protocol B

We next propose a variant of Protocol A; we call it Protocol B. The main idea behind this protocol is to reduce the number of shuffles by using the copy protocol instead of the verification protocol. Protocol B proceeds as follows.

1. The prover P places a single card on each cell according to the solution (as described in Section 3.2.1).

2. The verifier V regards n cards placed on each row as a commitment and applies the copy protocol (described in Section 3.2.3) to the commitment. After doing so, they obtain two identical copied commitments corresponding to each row. Further, verification of each row has been completed.

3. Note that each cell now has two identical cards. As in a similar way to GNPR Protocol 3, make $2n$ packets corresponding to the n columns and n subgrids. Each packet is shuffled. V opens all the packets and checks that each packet includes all numbers from 1 to n.

After Step 2 of this Protocol B, two cards are placed on each cell, and then, we need n additional cards to apply the copy protocol. Thus, this protocol needs $2n^2 + n$ cards in total. Next, let us count the number of shuffles in this protocol. Every execution of the copy protocol requires two pile-scramble shuffles, and it is performed for each of the n rows. The packet shuffle in Step 3 is performed once for each column and each subgrid. Therefore, the total number of shuffles is $4n$.

3.2.6. Correctness of the Proposed Protocols

In this subsection, we show that Protocol A described in Section 3.2.4 satisfies the properties of zero-knowledge proof. Since the proof of the validity of protocol B is almost the same as that of Protocol A, it is omitted.

Completeness: A prover P who knows the solution can place cards so that each row, each column and each subgrid contains all numbers. Whether the format of each row and each column is correct can be checked by the verification protocol. Further, V is convinced that P's input is not a solution to another puzzle instance by seeing the face-up cards having the same value as the filled-in cells.

Extractability: If P's input is invalid, at least one of rows, columns, and subgrids does not contain all numbers. Since such a packet can be always detected in the verification phase, V can always detect an illegal P's input. Therefore, Protocol A is perfectly extractable.

Zero-knowledge: In order to prove this, it is sufficient to show that all distributions of opened cards can be simulated by a simulator $M(\mathcal{I})$ who does not know the solution.

- In Steps 4 and 6 in the verification protocol, the commitments to rv and $r'r$ are opened. Their orders are uniformly distributed on S_n due to the pile-scramble shuffles. Thus, it can be simulated by $M(\mathcal{I})$.
- In Step 4 in Protocol A, the n packets corresponding to the n subgrids are opened. Their order is uniformly distributed on S_n due to the shuffle. Thus, it can be simulated by $M(\mathcal{I})$.

3.3. Proposed Protocol C for Sudoku: Interactive Inputs

In the previous section, we designed two zero-knowledge proof protocols for Sudoku, which have no extractability error thanks to the copy/verification technique. In this section, we consider more direct variant of GNPR Protocol 3.

Remember why GNPR Protocol 3 has a non-zero extractability error; if we guaranteed that the three cards placed on each cell were identical, we could immediately eliminate any extractability error, i.e., we could attain perfect extractability. Taking this in mind, we now consider an interaction between P and V to prepare input cards, so that every pile of three cards placed on each cell are identical. Our Protocol C, which we can call the *interactive input protocol*, proceeds as follows.

1. The verifier V publicly generates n piles each containing three "1" cards and turns their faces down; similarly, V generates n piles for each of 2 to n. After that, there are n^2 piles in total on the table.
2. P applies a pile-scramble shuffle to all the n^2 piles.
3. P picks one pile and looks at the cards in his/her hands to see the number on them while keeping it hidden from V. Then, P places the picked pile on a cell compatible with the solution.
4. P applies the same procedure as Steps 3 to all the remaining piles. After that, three cards are placed on each cell.
5. Execute Steps 2 to 4 of GNPR Protocol 3.

This is Protocol C. Because all three cards placed on each cell are guaranteed to be identical, Protocol C has no extractability error. Its performance is shown in Table 3.1.

3.4. Slitherlink

Hereinafter, we focus on constructing a ZKP protocol for *Slitherlink* that was introduced in 1989 in issue the 26th of Nikoli's Puzzle Times. It is also known as *Loop-the-Loop*. It is explained on Nikoli's web site as follows: "*Getting the loop right is absorbing and addictive. Watch out not to get lost in Slitherlink. It's amazing to see how endless patterns can be made using only four numbers (0, 1, 2 and 3)*". Slitherlink was proven to be NP-complete in [84] and other variants in [32].

Slitherlink is one of the most famous pencil puzzles published in the puzzle magazine *Nikoli*. The puzzle instance consists of lattice-like dots

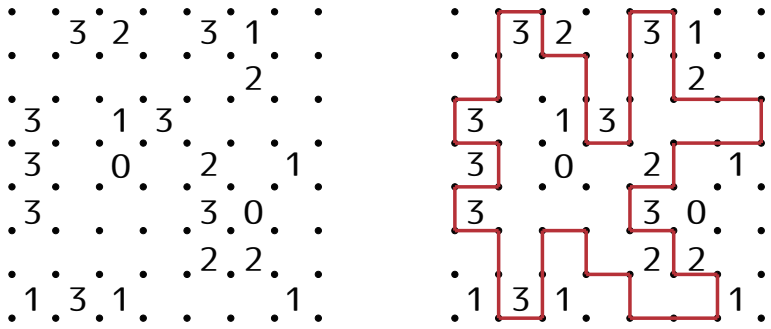


Figure 3.2: Example of a standard Slitherlink challenge, and its solution.

where some squares contain numbers between 0 and 3. The goal of the puzzle is to draw lines that satisfy the following rules[i]:

1. Connect vertical/horizontal adjacent dots with lines to make a single loop.

2. Each number indicates the number of lines that should surround its square, while empty squares may be surrounded with any number of lines.

3. The loop never crosses itself and never branches off.

Figure 3.2 shows an example of a Slitherlink puzzle and its solution; one can easily verify that all conditions are satisfied.

We introduce a new technique to construct a ZKP protocol for a puzzle where constructing a single loop is one of the requirements of the solution. The difficulty is to avoid leaking any information regarding the solution to the verifier. For this, we use a topological point of view; more precisely, we use the notion of homology that defines and categorizes holes in a manifold. The main idea is that after any continuous transformations, the number of holes always remains the same. Using this simple idea, we construct transformations that preserve the number of loops in the solution. First, the verifier checks that the initial configuration has only a single big loop. Then, by transforming in several steps this trivial big loop into the solution, the prover convinces step after step that the solution has only one loop at the end by proving that the transformation does not break the loop or introduce an extra hole. This construction is applied to Slitherlink in this article but it can be used for any other puzzles that require such type of features in their rules.

[i] https://www.nikoli.co.jp/en/puzzles/slitherlink.html

3.4.1. Notations

We use the following physical cards: $\boxed{\clubsuit}$ $\boxed{\clubsuit}$ $\cdots$ $\boxed{\heartsuit}$ $\boxed{\heartsuit}$; the black $\boxed{\clubsuit}$ and red $\boxed{\heartsuit}$ cards are called *binary cards* in this chapter. The backs of all cards are identical and denoted by $\boxed{?}$. In our construction, binary cards are used to encode the existence of a line while number cards are used for rearranging the positions of cards, as shown later.

Encoding. We encode Boolean values with two binary cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. Two face-down cards encoding 0 and 1 are called a *0-commitment* and a *1-commitment*, which are denoted by $\boxed{0\!\!|\!\!|}$ and $\boxed{1\!\!|\!\!|}$, respectively.

In our protocol, a 0-commitment placed on a gap between two adjacent dots means that there is no line on the gap, and a 1-commitment means that there is a line on the gap. With this encoding, we can represent a loop that is made of several lines. Note that given an x-commitment for $x \in \{0, 1\}$, swapping the two cards consisting the commitment results in an $\overline{x}$-commitment; thus, negation can be easily done.

Shuffle. Given a sequence of m face-down cards $(c_1, c_2, \ldots, c_m)$, a *shuffle* results in a sequence $\left(c_{r^{-1}(1)}, c_{r^{-1}(2)}, \ldots, c_{r^{-1}(m)}\right)$, where $r \in S_m$ is a uniformly distributed random permutation and S_m denotes the symmetric group of degree m.

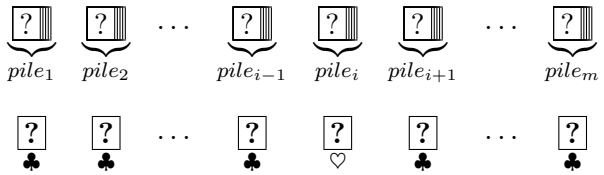
Pile-shifting Shuffle. The goal of this operation, which is also called pile-shifting scramble [61], is to *cyclically* shuffle piles of cards. That is, given m piles, each of which consists of the same number of face-down cards, denoted by $(pile_1, pile_2, \ldots, pile_m)$, applying a pile-shifting shuffle results in $(pile_{s+1}, pile_{s+2}, \ldots, pile_{s+m})$:

$$\underbrace{\boxed{?\!\!|\!\!|}}_{pile_1}\ \underbrace{\boxed{?\!\!|\!\!|}}_{pile_2}\ \cdots\ \underbrace{\boxed{?\!\!|\!\!|}}_{pile_m} \rightarrow \underbrace{\boxed{?\!\!|\!\!|}}_{pile_{s+1}}\ \underbrace{\boxed{?\!\!|\!\!|}}_{pile_{s+2}}\ \cdots\ \underbrace{\boxed{?\!\!|\!\!|}}_{pile_{s+m}},$$

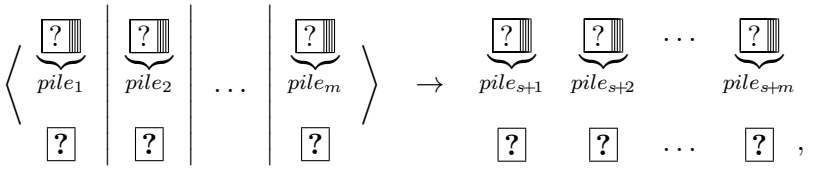
where s is uniformly and randomly chosen from $\mathbb{Z}/m\mathbb{Z}$. To implement pile-shifting shuffle, we use physical cases that can store a pile of cards, such as boxes and envelopes; a player (or players) cyclically shuffle them by hand until nobody traces the offset. It can be done by physical object as the one created for the physical ZKP for Sudoku in [72].

Chosen Pile Cut. It was proposed in [30]. *chosen pile cut* enables a prover to choose a pile $pile_i$ from m piles $(pile_1, pile_2, \ldots, pile_m)$ without revealing i to a verifier. The chosen pile cut proceeds as follows, given m piles along with m additional cards:

1. The prover P holds $m-1$ $\boxed{\clubsuit}$s and one $\boxed{\heartsuit}$. Then, P places m cards with their faces down below the piles such that only the i-th card is $\boxed{\heartsuit}$:



2. Regarding the cards in the same column as a pile, apply pile-shifting shuffle to the piles (denoted by $\langle \cdot | \ldots | \cdot \rangle$):



   where s is generated uniformly at random from $\mathbb{Z}/m\mathbb{Z}$ by this shuffle action.

3. Reveal all the cards in the second row. Then, one $\boxed{\heartsuit}$ appears, and the pile above the revealed $\boxed{\heartsuit}$ is $pile_i$, and hence, we can obtain the desired $pile_i$.

Owing to the pile-shifting shuffle in Step 2, revealing cards leaks no information about i and thus, chosen pile cut leaks no information about i, the index of the chosen pile.

3.5. Proposed Protocol for Slitherlink

In this section, we construct our physical zero-knowledge proof protocol for Slitherlink. The outline of our protocol is as follows.

Input Phase: The verifier V puts a 1-commitment (i.e., two face-down cards encoding 1) on every gap on the boundary of the puzzle board and 0-commitments on all the remaining gaps. In other words, V creates a single big loop whose size is the same as the board.

Topology-preserving Computation Phase: The prover P transforms the shape of the loop according to the solution. After this phase, V is convinced that the placement of 1-commitments satisfies Rules 1 and 3 of Slitherlink without the disclosure of any information about the shape.

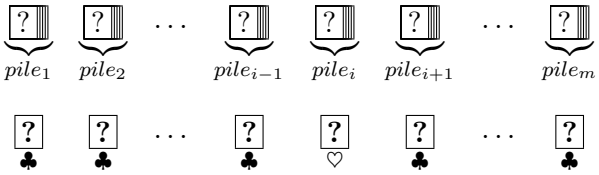
Verification Phase: V verifies that the placement of 1-commitments satisfies Rule 2 of Slitherlink.

We introduce some subprotocols in Section 3.5.1 before presenting our protocol in Section 3.5.2.

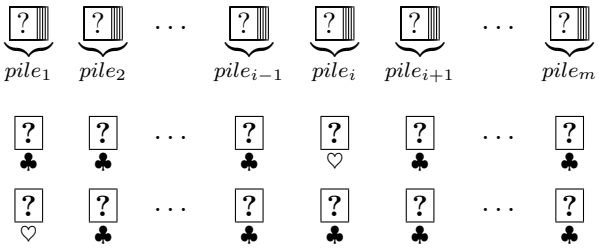
3.5.1. Subprotocols

Chosen pile protocol. This is an extended version of the chosen pile cut [30] explained in Section 3.4.1. Given m piles with $2m$ additional cards, this protocol enables P to choose the i-th pile and regenerate the original sequence of m piles.

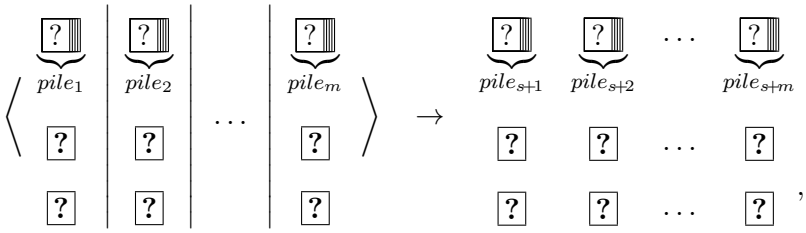
1. Using $m-1$ $\boxed{\clubsuit}$s and one $\boxed{\heartsuit}$, the prover P places m cards with their faces down below the given piles such that only the i-th card is $\boxed{\heartsuit}$:



   We further put m cards below the cards such that only the first card is $\boxed{\heartsuit}$:



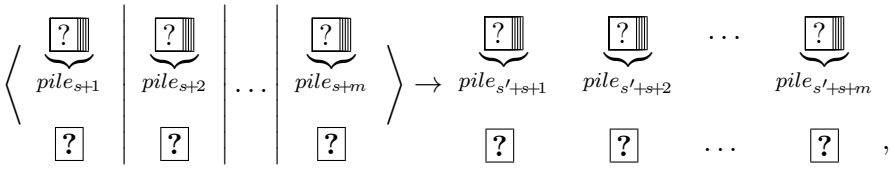
2. Considering the cards in the same column as a pile, apply a pile-shifting shuffle to the sequence of piles:



   where s is generated uniformly at random from $\mathbb{Z}/m\mathbb{Z}$.

3. Reveal all the cards in the second row. Then, one $\boxed{\heartsuit}$ appears, and the pile above the revealed $\boxed{\heartsuit}$ is the i-th pile (and hence, P can obtain $pile_i$). When this protocol is invoked, certain operations are applied to the chosen pile. Then, the chosen pile is placed back to the i-th position in the sequence.

4. Remove the revealed cards, i.e., the cards in the second row. Then, apply a pile-shifting shuffle:



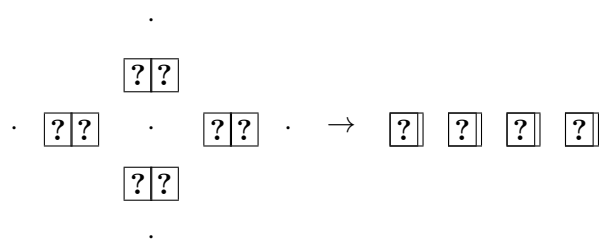
   where s' is generated uniformly at random from $\mathbb{Z}/m\mathbb{Z}$.

5. Reveal all the cards in the second row. Then, one $\boxed{\heartsuit}$ appears, and the pile above the revealed $\boxed{\heartsuit}$ is $pile_1$. Therefore, by shifting the sequence of piles, we can obtain a sequence of piles whose order is the same as the original one without revealing any information about the order of input sequence.

Verifying-degree Protocol. This protocol can verify that the "degree" of a target vertex (dot) is not four. Here, *degree* means the number of 1-commitments placed around a target vertex. Thus, the prover P wants to prove that there is at least one 0-commitment around the target vertex (when only P knows what the four commitments around the target are).

The verifying-degree protocol proceeds as follows.

1. Given four commitments that are placed around the target vertex, these can be regarded as a sequence of 4 commitments:



2. By using the chosen pile protocol, P chooses one of the 0-commitments. Open the chosen pile to show that it is 0. Now, V is convinced that the degree of the target vertex is not four. Then, V turns over all the opened cards. Because only a 0-commitment is always opened, no information about the four commitments is disclosed.

3. V performs the remaining steps in the chosen pile protocol. Then, all the cards are placed back to their original positions.

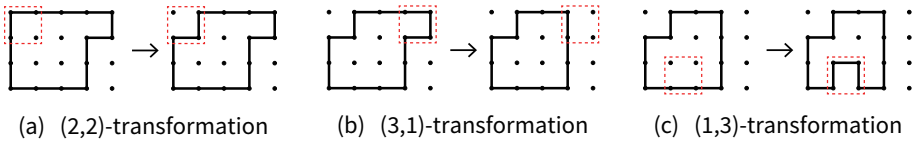
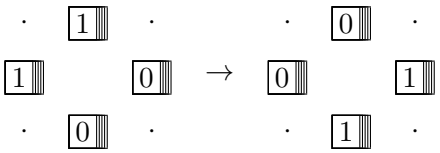


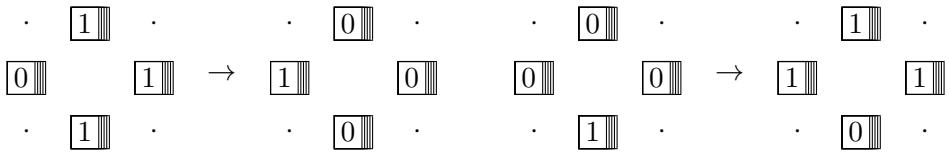
Figure 3.3: Three transformations.

Topology-preserving Computation. This protocol changes a given loop into another loop by one of the three transformations given in Figure 3.3. Each transformation changes the lines surrounding a square, represented by dash line in Figure 3.3.

Remember that a line is expressed by a commitment (i.e., two face-down binary cards) in our protocol. Therefore, for example, a (2,2)-transformation means



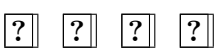
This can be implemented by swapping two cards of each commitment. (Remember that swapping the two cards performs negation of a commitment.) A (3,1)-transformation and a (1,3)-transformation can also be implemented by swapping two cards of each commitment:



Now, P wants to apply one of the three transformations while the applied transformation is hidden from V. Furthermore, P needs to show that the commitments around a target square are "transformable." Note that the three transformations are applicable to four commitments around a square if and only if there exists a 0-commitment facing a 1-commitment.

The topology-preserving computation proceeds as follows.

1. Pick four commitments around a target square:



2. P chooses a 0-commitment facing a 1-commitment using chosen pile protocol.

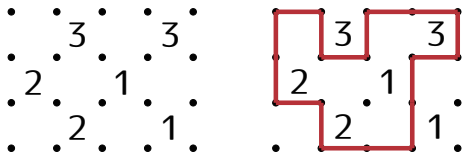


Figure 3.4: Small example of Slitherlink challenge, and its solution.

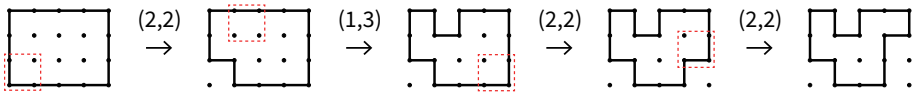
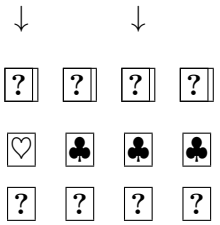


Figure 3.5: Transformation process.

3. V reveals the chosen commitment and the commitment that is two piles away from it:



Then, V checks that the two commitments are a 0-commitment and a 1-commitment to be convinced that any transformation can be applied.

4. After turning over all the opened cards, V performs the remaining steps in the chosen pile protocol to place all the cards back to their original positions.

5. Swap the two cards of each of the four commitments. (Remember that this results in negating all the four commitments, and hence, a transformation is applied.)

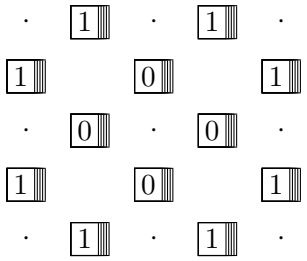
6. V applies the verifying-degree protocol to each of the four dots of the target square. Then, V is convinced that no dots of degree four have been obtained as the result of transformation. This guarantees that the loop was not split and thus, it remains a single loop.

3.5.2. Our Construction

As mentioned at the beginning of this section, the main idea behind our protocol is that the verifier V first creates a big loop and then the prover P transforms the loop into the solution loop one by one. Let us consider a puzzle instance shown in Figure 3.4 as an example. Our protocol transforms the loop as illustrated in Figure 3.5.

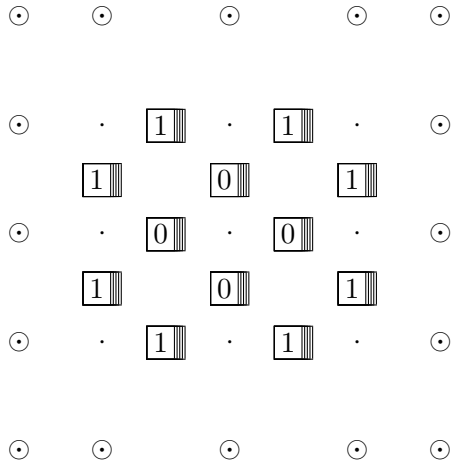
We are now ready to present the full description of our zero-knowledge proof protocol for Slitherlink.

Input Phase. The verifier V puts a 1-commitment on every gap on the boundary of the puzzle board and 0-commitments on all the other gaps. This placement corresponds to the single loop with the same size as the board. The following is an example of the placement of (2×2)-square puzzle board:



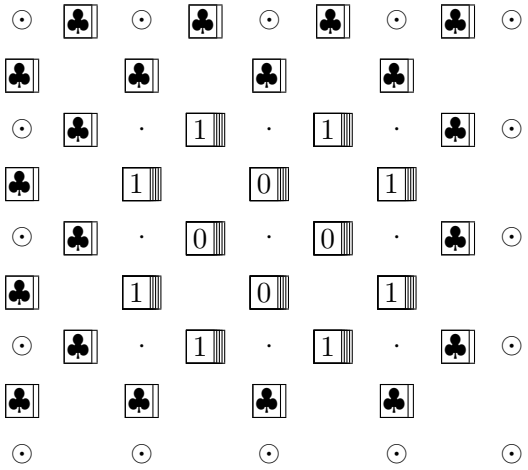
P will apply the topology-preserving computation to these commitments to transform the shape of the loop into the solution. Here, P needs to hide the target square. Therefore, we make a sequence of piles from the placed cards, pick the four target commitments using the chosen pile protocol, and apply the topology-preserving computation. To properly pick the four commitments, a sequence of piles is formed, as follows.

We first expand the puzzle board by adding dots around the original board. (For explanation, the expanded dots are denoted by $\odot$.)

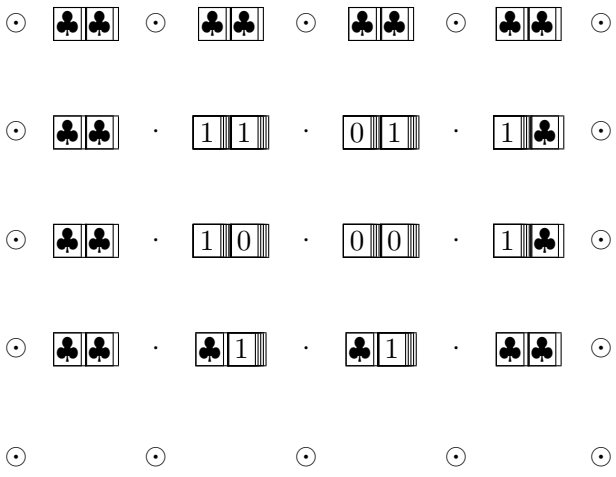


Note that the expanded area is unrelated to the actual puzzle board. V puts dummy commitments on the gaps at the expanded area other than the right and the bottom ends. Each dummy commitment consists of two black cards $\boxed{\clubsuit}\boxed{\clubsuit}$ to prevent the loop from spreading over the expanded area. We

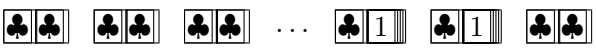
denote the dummy commitment by $\boxed{\clubsuit}$.



Next, V makes a sequence of 4-card piles as follows. For each square, V first makes a pile from the commitments placed on the left and the top (the commitment on the gap between each vertically consecutive dots is placed on the commitment on its upper right.)



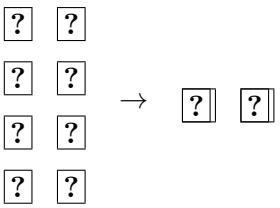
Then, pick 4-card piles from top to bottom:



to make a sequence of piles:

Topology-preserving computation phase. In this phase, P applies transformations (explained in Section 3.5.1) to stepwise change the big loop to the solution loop. Let n be the size of the puzzle instance, namely the number of squares on the puzzle board. Then, note that P can make the solution loop by at most n transformations.

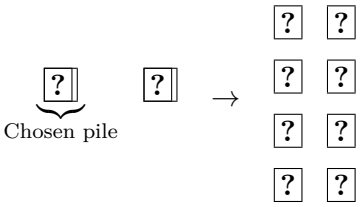
1. P applies the following exactly $n-1$ times such that either the resulting loop is already the solution, or one more transformation will end up the solution. (This is possible because successive two transformations (of the same) to the same square keep the loop unchanged.)
   (a) P applies the chosen pile protocol to the sequence of 4-card piles: P picks a 4-card pile composed of left and top edges of the square that P wants to transform. The other edges can be picked by counting the distance from the chosen pile[ii].
   (b) P applies the topology-preserving computation to the four picked commitments.
   (c) V performs the remaining steps in the chosen pile protocol to place the cards back to their original positions.
2. P applies one more transformation or does not change the solution loop so that V does not learn which action occurs, as follows.
   (a) Similarly to Step 1 (a) above, P picks four commitments around the target square.
   (b) By using the method explained in the topology-preserving computation, V confirms that any transformation is applicable.
   (c) V arranges the four commitments vertically and makes a pile from each column:



   Note that swapping two piles results in inverted value of each commitment. Thus, it is equivalent to applying a transformation.
   (d) Using the chosen pile cut, if P wants to transform the target square, then P chooses the right pile; otherwise, the left pile is chosen.

[ii] In the above example, the bottom edge corresponds to the pile which is 4 piles away from the chosen pile. Note that the distance between any two piles never changes because only Pile-Shifting Shuffle is applied.

(e) Rearrange the cards vertically such that the chosen pile is placed at left:



(f) V makes four commitments from each row, performs the remaining steps in the chosen pile protocol, and places each commitment back to their original position.

3. Finally, all cards are placed on the puzzle board and the cards at the dummy area are removed.

Verification phase. V is now convinced that the placement of 1-commitments is a single loop (Rule 1) and it never branches off (Rule 3). Therefore, V only needs to verify that the placement satisfies Rule 2 of Slitherlink.

Now, V verifies that the number on each square is equal to the number of lines surrounding it. The verification proceeds as follows, where we virtually assume that the board is colored like a checkered pattern so that all squares in the first row are alternation of blue and yellow, those in the second row are alternation of yellow and blue, and so on.

1. V picks all left cards (if the square is virtually blue) or all right cards (if the square is yellow) of four commitments around a square on which a number is written:

 .

2. P shuffles the four cards.

3. V reveals the four cards.

   - If V picked all the left cards of four commitments in Step 1, V checks that the number of red cards is equal to the number on the square.
   - If V picked all the right cards of four commitments in Step 1, V checks that the number of black cards  is equal to the number on the square.

4. Apply Steps 1 to 3 to all other numbered squares. (Note that a commitment is related to at most one blue numbered square and one yellow numbered square.)

Our protocol uses $6(p+2)(q+2)+8$ cards in total, where we have a $p \times q$ board.

3.6. Security Proofs for Our Construction

In this section, we show that our construction presented in Section 3.4 satisfies the completeness, extractability, and zero-knowledge properties.

Completeness. In the input phase, V is convinced that 1-commitments are placed in a single loop because V does the operations by himself/herself, and hence, V is convinced that the placement satisfies Rules 1 and 3 of Slitherlink. As explained in Section 3.5.1, the transformations are applied to only applicable squares. Thus, every transformation is performed while preserving Rules 1 and 3. By verifying that the placement satisfies Rule 2 in verification phase, V is convinced that P knows the solution. Therefore, if P has a solution for the puzzle then P can always convince V.

Remember that P uses only (3,1)-, (1,3)-, and (2,2)-transformations in the topology-preserving computation to transform a single loop into the shape of the solution. We now prove that this is possible in Theorem 1.

Theorem 1 *Let n be the number of squares in the puzzle instance (namely, the big loop), and let k be the number of squares inside its solution loop. By applying a transformation to the loop exactly $n - k$ times, the big loop can be transformed into the solution loop.*

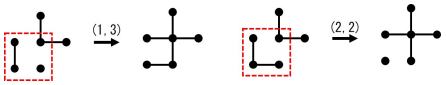
To prove Theorem 1, we first show Lemmas 1 and 2.

Lemma 1 *The resulting placement of 1-commitments after the topology-preserving computation always represents a single loop.*

Proof *Remember Steps 2 and 6 in the topology-preserving computation: Due to Step 2, the target square is guaranteed to be none of the following two ones (up to rotations).*

That is, one of (2,2)-, (3,1)-, and (1,3)-transformations is always applied to the target square.

Due to the execution of the verifying-degree protocol in Step 6, the following two transformations that make a loop split cannot occur.

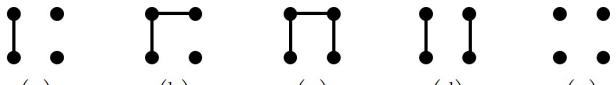


Therefore, it remains a single loop. □

Lemma 2 *For any single loop, there is always a (3,1)-, (1,3)-, or (2,2)-transformation that increases the number of squares inside the loop by exactly one.*

Proof *Consider a single loop; let ℓ be the number of squares inside the loop. To prove this lemma, we show that there always exists a square on the board such that a (3,1)-, (1,3)-, or (2,2)-transformation can be applied to the square such that ℓ increases. Note that the loop remains single after the application of the transformation by Lemma 1.*

If $\ell \leq 2$, a (1,3)-transformation increases the number of squares by one. Thus, one may assume that $\ell \geq 3$. Then, any square outside the loop can be classified in one of the following five types (up to rotations):



If none of (a), (b), and (c) exists, all squares outside the loop are either (d) or (e), and hence, it would not be a sigle loop. Therefore, at least one square of type (a), (b), or (c) must exist outside the loop.

Applying a (3,1)-, (1,3)-, or (2,2)-transformation to such an external square results in increasing ℓ by one. □

By these lemmas, Theorem 1 can be proved.

Proof (of Theorem 1) *By Lemmas 1 and 2, we can always increase the number of squares inside the solution loop by a transformation. Therefore, we can repeat the transformation so that the solution loop becomes the big loop. This means that, conversely, the big loop can be transformed into the solution loop by applying (3,1)-, (1,3)-, or (2,2)-transformation exactly $n-k$ times.* □

Extractability. Only the person who knows the solution can transform the loop so that the shape satisfies Rule 2. Therefore, V can detect any illegal prover in Verification Phase. Thus, if the prover does not know the solution for a puzzle, then V will be never convinced, irrespective of P's behavior.

More formally, to prove the extractability, we are required to show that any shape that does not satisfy Rule 1, 2, or 3 is always rejected during the protocol.

Theorem 2 *If the prover does not know the solution for the Slitherlink puzzle, then the verifier always rejects regardless of the prover's behavior.*

To prove Theorem 2, we show that the resulting loop after the topology-preserving computation always satisfies Rules 1 and 3 (as in Lemma 1) and any single loop that does not satisfy Rule 2 is always rejected in Verification Phase (as in Lemma 3). Therefore, any single loop except for the solution is always rejected.

Lemma 3 *Any (single) loop that does not satisfy Rule 2 is always rejected in Verification Phase.*

Proof *Consider any (single) loop that does not satisfy Rule 2, i.e., there are four commitments surrounding a numbered square such that the number of 1-commitments among them is not equal to the number. Due to Step 3 in Verification Phase, all the left (or right) cards of four commitments are turned over (after shuffling them), and hence, the number of 1-commitments is revealed. This means that the verifier can always reject any (single) loop that does not satisfy Rule 2.* □

Proof (of Theorem 2) *By Lemma 1, the resulting loop after the topology-preserving computation is always single, i.e., it satisfies Rules 1 and 3. By Lemma 3, if it does not satisfy Rule 2, the verifier always rejects it in Verification Phase. That is, any loop except for the solution cannot go through Verification Phase.* □

Zero-knowledge. In our construction, all the opened cards have been shuffled before being opened. Therefore, all distributions of opened cards can be simulated by a simulator $M(\mathcal{I})$ who does not know the solution. For example, at Step 3 in Verification Phase, the Pile-Scramble Shuffle have been applied to opened commitments; thus, this is indistinguishable from a simulation putting randomly 1-commitments such that the number of them is equal to the number of the square.

3.7. Conclusion

In this chapter, we proposed three card-based zero-knowledge proof protocols for Sudoku. See Table 3.1 again for a comparison of performances of GNPR Protocol 3 and our protocols; all our protocols have no extractability error and use fewer cards, and hence, they are efficient.

Moreover, we introduced a new technique that can transform a single loop encoded with physical objects into a new geometrical figure while preserving the single loop. By using this secure computation, we constructed the first physical zero-knowledge proof protocol for Slitherlink.

4. Secure Implementations

It is known that by using a deck of cards, we can realize secure multiparty computations. For example, consider a secure AND computation of bits $a \in \{0, 1\}$ and $b \in \{0, 1\}$, i.e., assume that we only want to know the value of $a \wedge b$. By utilizing a black card $\boxed{\clubsuit}$ and a red card $\boxed{\heartsuit}$, we can represent the value of a bit as follows:

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0, \quad \boxed{\heartsuit}\boxed{\clubsuit} = 1.$$

According to this encoding, each of the input bits a and b can be represented using two face-down cards of different colors:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\ \underbrace{\boxed{?}\boxed{?}}_{b}.$$

A pair of face-down cards (such as in the above example) is called a *commitment*. That is, the two cards on the left constitute a commitment to a, and the two cards on the right constitute a commitment to b. As in this example, the cards we use are either black cards $\boxed{\clubsuit}$ or red cards $\boxed{\heartsuit}$, whose backs are assumed to be identical $\boxed{?}$. As shown in Table 4.1, many protocols have been designed to perform a secure AND computation, from among which we introduce the Mizuki–Sone AND protocol [48]. Given commitments to inputs a and b along with two additional cards $\boxed{\clubsuit}\boxed{\heartsuit}$, the protocol works as follows.

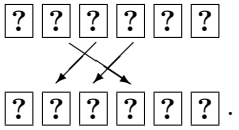
1. A commitment to 0 is placed between the two input commitments:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{b} \ \rightarrow\ \underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{b}.$$

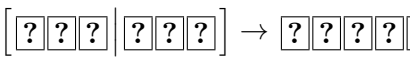
Table 4.1: Some committed-format AND protocols (RC: Random Cut, RBC: Random Bisection Cut)

	#Colors	**#Cards**	**Type of Shuffles**	**Avg. #Trials**
[9]	4	10	RC	6
[56]	2	12	RC	2.5
[78]	2	8	RC	2
[48]	2	6	RBC	1
[2]	2	5	RC,RBC	5

2. The sequence order is then rearranged as follows:

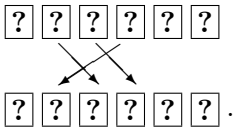


3. A *random bisection cut* is applied as follows:

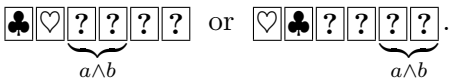


   A random bisection cut is a shuffling operation that bisects a sequence of cards and swaps the two halves randomly. Therefore, the shuffle results in two possible cases, depending on whether the two halves are swapped, each with a probability of 1/2.

4. The sequence order is rearranged as follows:



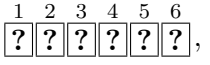
5. The two left-most cards are turned over, and we are able to obtain a commitment to $a \wedge b$ as follows:



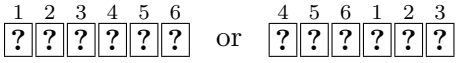
Although we omitted an explanation regarding the correctness and secrecy of this protocol, one can confirm that the protocol outputs a commitment to $a \wedge b$ by using six cards after one execution of the random bisection cut [48]. (A protocol such as this that outputs commitments is called a *committed-format protocol*.)

Random Bisection Cut

In practice, humans can perform a random bisection cut by shuffling the two halves after bisecting a given sequence of cards, as illustrated in Figure 4.1. Thus, given a sequence of six cards



a random bisection cut results in



(a) Bisection of a sequence of cards (b) Shuffling of the two halves

Figure 4.1: Execution of a random bisection cut

with a probability of 1/2 for each possibility, where the numbers attached to the cards are for the sake of convenience.

Following the computational model formalized in the studies [31,45], this random bisection cut can be described as follows:

$$(\mathsf{shuffle}, \{\mathsf{id}, (1\,4)(2\,5)(3\,6)\}),$$

where id represents the identity permutation, and an expression, such as $(1\,4)$, represents a cyclic permutation. Therefore, id indicates that the two halves are not swapped, and permutation $(1\,4)(2\,5)(3\,6)$ indicates that the two halves are swapped.

Historically, random bisection cuts first appeared when a six-card AND protocol was designed in 2009 [48]. Even before this design, some committed-format AND protocols had been designed. These earlier protocols employed *random cut* as a shuffling operation, as shown in Table 4.1. A random cut refers to a cyclic shuffling operation. For example, given eight face-down cards

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array},$$

a random cut results in one of the following eight cases, each with a probability of 1/8:

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array}$$

$$\begin{array}{cccccccc} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 1 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array}$$

$$\vdots$$

$$\begin{array}{cccccccc} 8 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} \end{array}.$$

Therefore, by following the computational model in the studies [31,45], we can similarly describe the random cut as

$$(\mathsf{shuffle}, \{\mathsf{id}, \pi, \pi^2, \pi^3, \pi^4, \pi^5, \pi^6, \pi^7\}),$$

Table 4.2: Some other protocols

	#Colors	#Cards	Type of Shuffles	Avg. #Trials
◦ *Non-committed-format AND Protocols*				
[11]	2	5	RC	1
[44]	2	4	RBC	1
◦ *Committed-format XOR Protocols*				
[9]	4	14	RC	6
[49]	2	10	RC	2
[48]	2	4	RBC	1

where $\pi = (8\,7\,6\,5\,4\,3\,2\,1)$.

As seen in Table 4.1, committed-format AND computations have become more efficient by virtue of the introduction of the random bisection cut in 2009. This introduction also provided the additional benefit of improving the efficiency of non-committed-format AND computations and committed-format XOR computations, as detailed in Table 4.2. In addition, other efficient protocols have been designed using random bisection cuts [24, 38, 40, 58, 59].

Contribution

As explained earlier, *card-based protocols* are intended in practice to be executed by humans who actually desire to perform secure multiparty computations by using a real deck of cards. Hence, when we execute a card-based protocol, it is expected that all players gather at the same physical location, and perform operations, such as shuffles, in public, as in the case of ordinary card games [46] [i].

To implement a random cut in such a situation, it is sufficient that each player cuts a sequence of face-down cards in turn until all players are satisfied with the result. Indeed, in practice, it is relatively easy to implement a random cut such that nobody is able to determine the result of the shuffle at all. We will discuss this further in Section 4.4.

When players operate a random bisection cut, if they are not familiar with playing cards and have difficulty in shuffling the two halves such that

[i] It should be noted that recent work (e.g., [53, 54]) considers the use of private actions by players to design efficient protocols.

Figure 4.2: Each half is placed in an envelope

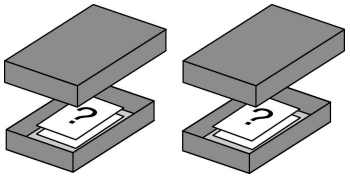


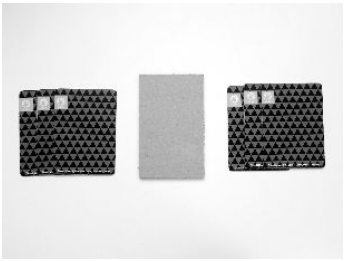
Figure 4.3: Each half is placed in a box

each half stays together, as in Figure 4.1(b), then they may secure each half using paper clips or envelopes [44,48]. By using these auxiliary tools, we are able to fix each of the two halves together, as shown in Figure 4.2. Following this, it suffices to swap the two bundles of cards randomly. However, the result of the shuffle could be revealed when we execute a random bisection cut in public because there are only two possibilities, i.e., the two halves of the card sequence are swapped or not swapped. That is, someone may count how many times the two bundles are swapped. To avoid such a leak of information, one solution is that each player shuffles the two bundles behind his/her back or under a table, so that other players cannot see whether the two bundles are swapped. In this case, it may be preferable to use envelopes or boxes (as illustrated in Figures 4.2 and 4.3) rather than paper clips to avoid malicious actions[ii]. Nevertheless, it is desirable for all actions to be performed in front of all players and/or third parties publicly. Therefore, in Sections 4.1, 4.2, and 4.3, we present implementations of random bisection cuts, where every action can be performed in public.

Outline

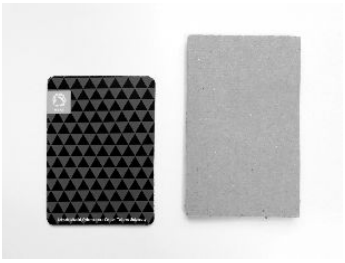
The remainder of this chapter is organized as follows. In Section 4.1, we present some methods for implementing a random bisection cut by using auxiliary tools. In Section 4.2, we propose a method to reduce the execution of a random bisection cut to the execution of random cuts using dummy cards. In Section 4.3, we propose another method to implement a random bisection cut without relying on dummy cards. In Section 4.4, we discuss secure implementations of the random cut.

[ii]Envelopes or boxes can be also used for implementing other types of shuffling operations ([21,23,24,63]).



(a) Separator and two halves

(b) One half is placed on the separator



(c) The pile consisting of one half and the separator is flipped

(d) The other half is placed on the pile

Figure 4.4: Setup for spinning throw

4.1. Executing a Random Bisection Cut Using Auxiliary Tools

In this section, we provide methods for implementing a random bisection cut by using auxiliary tools that consist of everyday objects.

4.1.1. Use of a Separator Card and Rubber Band

In this subsection, we present a novel method of performing a random bisection cut by using a separator card with a rubber band. Both sides of the separator (as shown in the middle of Figure 4.4(a)) must be indistinguishable.

The method works as follows. First, a sequence of cards is bisected, with one half placed on the separator, as shown in Figure 4.4(b). Second, the pile consisting of one half and the separator is turned over, as shown in Figure 4.4(c)[iii]. Third, the other half is placed on the pile, as shown in Figure 4.4(d), and these are fixed together by using a rubber band to prevent the cards from scattering. Next, the pile is thrown in a spinning manner (as illustrated in Figure 4.5). We call this action a *spinning throw*. After the pile is caught, we are completely unsure of which half is on the top. Finally,

[iii] The separator prevents information regarding the color of cards from being leaked.

(a) Hold the pile of cards

(b) Throw the pile like a coin-flipping

Figure 4.5: A spinning throw

the rubber band is removed, and the actions described in Figure 4.4 are undone in the reverse order from Figures 4.4(d) to 4.4(a). In this manner, we can conduct a random bisection cut securely.

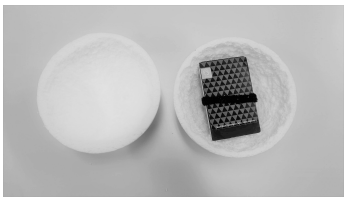
4.1.2. More Secure Implementation by Using a Ball

During execution of the spinning throw shown in Figure 4.5, the outcome cannot be traced by human eyesight. For checking the security of this shuffle, we recorded a video[iv] of a spinning throw and confirmed that we could not determine the result of the shuffle by watching the video even in slow-motion.

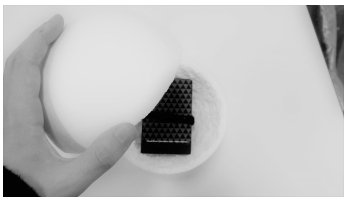
Nevertheless, someone may assume that if we use an enterprise high-speed camera, the result of this shuffle might possibly be revealed. To address this issue, we considered the use of a curving polystyrene foam ball, as shown in Figure 4.6.

The procedure of using the ball device is as follows. After banding the pile by using a rubber band and a separator, it is placed in one half of a curving ball, as illustrated in Figure 4.6(a), covered with the other half (Figure 4.6(b)), and the halves are then taped together (Figure 4.6(c)) so as not to be separated. Finally, the ball is thrown in the air in a spinning manner, as illustrated in Figure 4.7. In this case, the pile spins inside the ball, and is therefore shuffled out of sight of everyone present. Consequently,

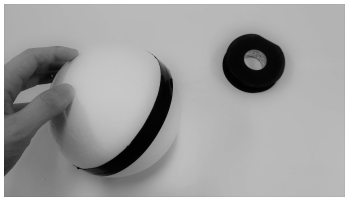
[iv] The camera we used was SONY FDR-AX40 and the video was recorded in 4K resolution and 60 fps.



(a) The pile is placed into one half of the curving ball



(b) Cover the pile with the other



(c) The ball's halves are taped together

Figure 4.6: Making a ball device to execute a spinning throw securely

a random bisection cut is implemented perfectly.

4.2. Execution of a Random Bisection Cut by Using Dummy Cards

In this section, we propose a method for reducing the execution of a random bisection cut to the execution of random cuts by using dummy cards.

Hereafter, we assume that we want to apply a random bisection cut to a sequence of $2n$ cards, where $n \geq 2$:

$$\Big[\underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}_{n \text{ cards}}\Big|\underbrace{\boxed{?}\boxed{?}\cdots\boxed{?}}_{n \text{ cards}}\Big].$$

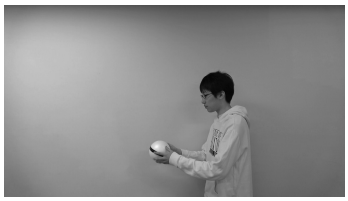
Formally, it cam be defined as

$$(\mathsf{shuffle}, \{\mathsf{id}, (1\ n{+}1)(2\ n{+}2)\cdots(n\ 2n)\}),$$

following the card-based computational model [31, 45].

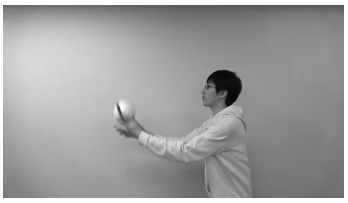
As dummy cards, we use cards with backs as $\boxed{?}$ and faces other than $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$, namely $\diamondsuit$ or $\spadesuit$. Specifically, we use $2\lceil s/2 \rceil$ $\diamondsuit$ and $2\lfloor s/2 \rfloor$ $\spadesuit$ cards, where $s \geq 2$. That is, we have a total of $2s$ additional cards.

By using such dummy cards, we are able to implement a random bisection cut as follows.



(a) Holding the ball

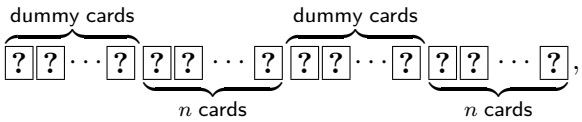
(b) Throwing the ball



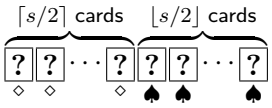
(c) Catching the ball

Figure 4.7: The scene of throwing the ball device in the air with a spin

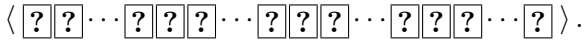
1. Place dummy cards with their faces down, as follows:



where the dummy cards are arranged as

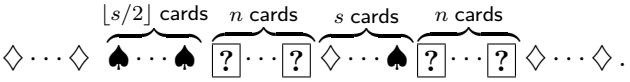


2. Apply a random cut:



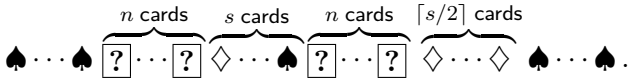
3. Turn over the left-most card.

   (a) If the face-up card is ♢, then turn over cards in the forward (the right-hand) direction until ⌊s/2⌋ ♠ cards appear. Now that the positions of all of the dummy cards have been determined, all of them can be removed:



   (b) If the face-up card is ♠, then turn over cards in the backward direction (aside from cyclic rotations) until ⌈s/2⌉ ♢ cards appear.

After determining the positions of all of the dummy cards, all of them can be removed:



(c) If the face-up card is ♣ or ♡, then turn it over again and return to Step 2.

In this manner, after all the dummy cards have been removed, a random bisection cut is completed.[v]

In Step 3, the probability that either (a) or (b) occurs is $s/(n+s)$. Therefore, we are able to implement a random bisection cut by using $2s$ dummy cards after an average of $(n+s)/s$ executions of the random cut.

This method of discarding dummy cards was first devised by Crépeau and Kilian [9], when they proposed some random permutation generating protocols. We adopted their idea in this study.

Regarding the parameter s, a trade-off exists between the number of required cards and the average number of executions of the random cut. For example, if we want to implement the Mizuki–Sone six-card AND protocol [48] with an average number of two random cuts, then we require six additional dummy cards. This requires more cards than Stiglic's eight-card AND protocol [78] (although the Mizuki–Sone six-card AND protocol might have the advantage that its correctness is simpler to understand).

4.3. Utilizing Vertical Asymmetricity of the Backs of Cards

In Section 4.2, we required additional types of cards to reduce the execution of a random bisection cut to the execution of random cuts.

In this section, we present another method to implement a random bisection cut without relying on dummy cards. To this end, we exploited the vertical asymmetricity of the backs of cards [?]. As the back is asymmetric, it can be seen as either [?] or [ʖ], depending on its position.

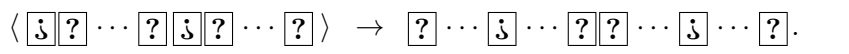
4.3.1. Reduction to a Random Cut

The method of reducing to a random cut is quite simple, described as follows.

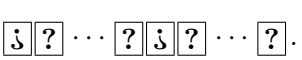
1. The first card of each half is rotated 180°:

[v]Note that we need two types of dummy cards (♢, ♠) to determine their exact positions.

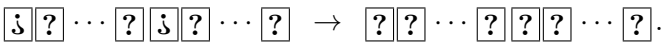
2. Apply a random cut:



Then, cyclically shift the sequence with the first card as $\boxed{\text{♣}}$:



3. Rotate the two of $\boxed{\text{♣}}$ again:



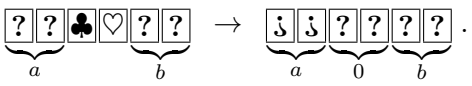
In this manner, by executing one random cut, we are able to implement a random bisection cut with no dummy card.

For example, the Mizuki–Sone six-card AND protocol [48] can be implemented using one random cut, as described in the following subsection.

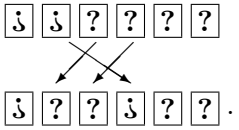
4.3.2. Six-Card AND Protocol with a Random Cut

We rewrite the six-card AND protocol [48] by using the method presented in Section 4.3.1.

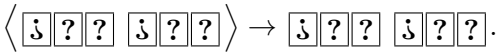
1. The two cards of a commitment to a are placed upside down:



2. The order sequence is rearranged as follows:



3. A random cut (rather than a random bisection cut) is applied, and then the card(s) is cyclically shifted so that the first card is $\boxed{\text{♣}}$:



4. Two of $\boxed{\text{♣}}$ are turned over:

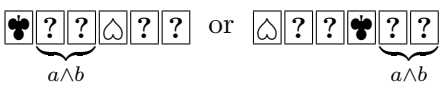


Figure 4.8: Simple execution of a random cut

4.3.3. Application to Pile-Shifting Scrambles

We can extend the method described in Section 4.3.1 to implement a more general shuffle, called the *pile-shifting scramble*. In a pile-shifting scramble, a card sequence of n cards (such that $n \mod m = 0$) is divided into m piles, and a random cut is applied to the sequence of the piles without changing the order of the cards inside each plie. Therefore, a random bisection cut is a special case of pile-shifting scrambles, i.e., it corresponds to $m = 2$. One can easily have a reduction of a pile-shifting scramble to a random cut based on the idea that the first card in each pile is marked by placing it upside down.

In our method, we must apply a random cut to cards with asymmetric backs, and hence information regarding the result of the shuffle could be leaked more easily than with cards that have identical backs.

By considering the aforementioned, we discuss secure implementations of the random cut in the next section.

4.4. Secrecy of Implementations of the Random Cut

In Sections 4.2 and 4.3, we proposed some methods for reducing the execution of a random bisection cut to the execution of random cuts. In general, it is believed that a random cut can be securely implemented by humans. To support this belief, we discuss the secure implementation of a random cut by shuffling a real deck of cards.

As a random cut consists of a cyclic shuffle, its simple implementation proceeds as in Figure 4.8: some cards (or a card) are taken from the top of the pile, and then moved to the bottom of the pile (this is called a *cut*). At every cut, we should change the number of cards to be moved. For such an implementation, some people can trace the move of cards.

Thus, we require an alternative secure implementation of the random cut. The point of strength of shuffle secureness is the visual observation of the number of cards moved at every cut, and summing up of all the numbers. Hence, the key is to ensure that it is impossible for people to count the number of cards moved during every cut.

One concept is to move cards from the bottom to the top of the pile

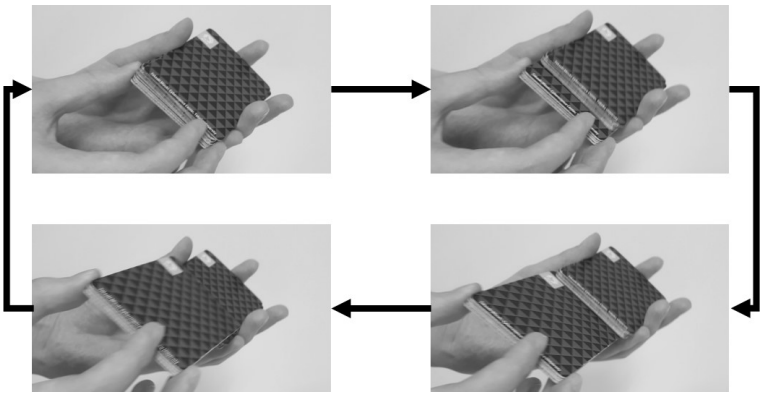


Figure 4.9: Execution of a "Hindu cut"; take a random number of cards from the bottom, move them to the top of the pile, and repeat this.

instead of moving them from the top to the bottom when executing a cut operation. As such, the recognition of the number of cards that have been moved becomes difficult. Moreover, if the positions of the cards are out of alignment, as shown in Figure 4.8, then it is possible to easily recognize the number of cards moved. Therefore, we should ensure that cards are not out of alignment when we execute cut operations.

Based on this concept, we found that a variation of the so-called Hindu shuffle (shown in Figure 4.9) is effective for preventing the cut operation from being revealed (we call this the *Hindu cut*).

We have experimentally demonstrated the security of this Hindu cut in the paper [81]. A summary of the experience is as follows. We requested 72 participants (who were non-specialists) to watch a video depicting the execution of the Hindu cut to the sequence of eight cards containing two non-identical back sides. As a result, 64 participants told us that they had not been able to track the move of the shuffle. Regarding the remaining 8 participants, to rule out wild guesses, we asked them to watch four more videos, and consequently, none of them was able to answer correctly for all of the five video. Refer to [81] for the details.

4.5. Conclusion

The random bisection cut has played an important role in improving card-based protocols. However, implementation issues have not been previously discussed. Therefore, in this chapter, we proposed some novel methods for implementing the random bisection cut and demonstrated that it could be implemented in practice. Users can choose one from our several methods proposed in Sections 4.1, 4.2, and 4.3, depending on the availability of auxiliary tools and the patterns of backs of available cards.

Part II.

Cryptographic Protocols with Everyday Tools

5. Balls and Bags

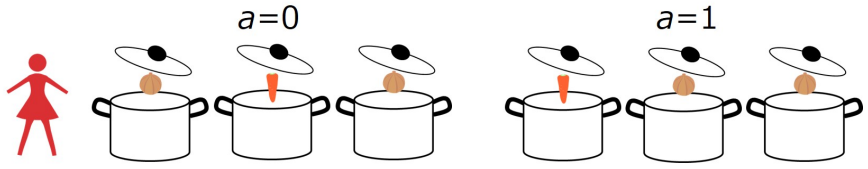
Three cryptographers are just cooking Borscht soup at the kitchen. Each of them has brought typical ingredients for Borscht soup such as carrots and onions. They might be paying for the ingredients, or some of them might be funded by NFSA (National Fictional Security Agency). The three cryptographers respect each other's ideology to have a relation to NFSA, but they wonder if they eat food funded by NFSA. All they have in the kitchen are the ingredients (namely, carrots and onions) and saucepans with Borscht soup. Then, they decide to resolve their uncertainty by conducting a secure AND computation with the ingredients and saucepans, to make sure whether they all paid or not. We call this the *Cooking Cryptographers Problem*, which is named in honor of the *Dining Cryptographers Problem* [7].

Cooking Cryptographers

For simplicity, let us consider the Cooking Cryptographers Problem with two players. Assume that Alice and Bob have private inputs $a, b \in \{0, 1\}$, respectively (the individual input is 0 if he/she was supported by NFSA; otherwise, it is 1). Our goal is to compute the two-input logical AND function $f(a, b) = a \wedge b$ without revealing any information except for the output value. Remembering that they are in the kitchen, let us construct a secure protocol for this function using cooking tools and ingredients. Interestingly, we can give an example of the computation using two carrots, four onions, and three saucepans, as follows.

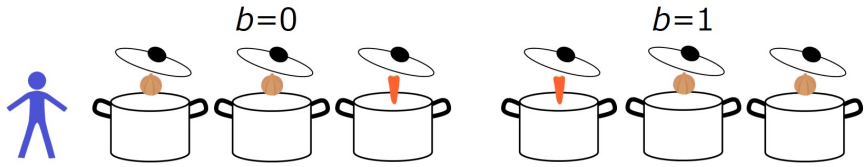
1. There are three saucepans filled with Borscht soup on kitchen countertops. Each of Alice and Bob holds one carrot and two onions.

2. If $a = 0$, Alice puts the carrot into the second saucepan and the onions into the other saucepans privately (so that Bob cannot learn which saucepan contains the carrot). If $a = 1$, she puts the carrot into the first saucepan and the onions into the other saucepans.

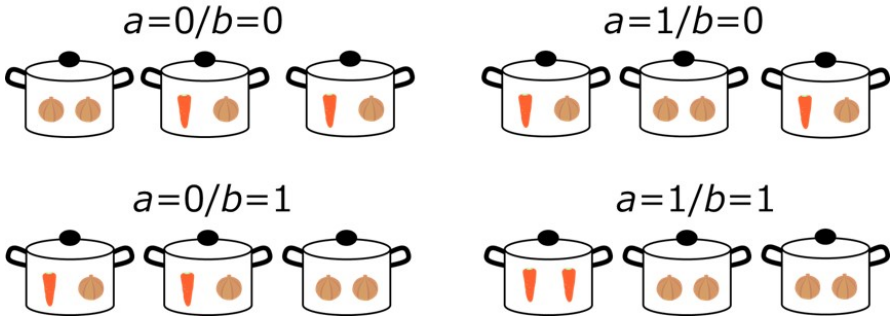


   The ingredients go to the bottom so that nobody sees them directly in any saucepans.

3. If $b = 0$, Bob privately puts the carrot into the third saucepan and the onions into the other saucepans. If $b = 1$, he puts the carrot into the first saucepan and the onions into the other saucepans.



Note that the two carrots are in the same saucepan if and only if $a = b = 1$.



4. Alice and Bob shuffle the three saucepans so that the resulting order of the three saucepans becomes unknown to them.

5. After simmering the Borscht soup in the three saucepans, they enjoy eating the cooked Borscht soup in the three saucepans; if there is a saucepan of Borscht soup only with carrots, we have $f(a, b) = a \wedge b = 1$ (meaning that none of Alice and Bob was supported by NFSA); otherwise, $a \wedge b = 0$.

Contribution

In this chapter, we formalize the above two-party protocol with a general setting. Because it is not so easy or realistic to cook Borscht soup whenever people want to perform a secure computation, let us replace ingredients and saucepans (with Borscht soup) by colored balls (such as red and white balls) and non-transparent bags, respectively. Assume that a bag possibly includes balls but the colors of the balls are invisible from the outside. Balls and bags are easy to prepare, and they are also familiar tools for learning Probability in high school. Therefore, basing on balls and bags will be more human-friendly than cooking soup for secure computations.

Then, our problem is reformulated as: *Can we realize secure multiparty computations (MPCs) [83] with such balls and bags?* As will be reviewed, there are MPC protocols with familiar tools, such as a deck of physical cards [9, 11, 44]. Unlike these existing protocols, we attempt to construct the first protocol using colored balls and bags together with simple actions: putting balls into a bag, shuffling the order of bags, and taking balls from a bag. Notice that, similar to ingredients in Borscht soup, balls in a bag have an interesting property that a collection of balls automatically becomes disordered once they are put into a bag, namely *automatic shuffle*.[i]

We positively answer the above question; we construct simple protocols to establish MPCs of the logical AND function with more than two inputs as well as general MPCs. We also give a formal treatment for our protocols and their security (namely, *ball-based cryptography*). To easily confirm correctness and security of a protocol, we construct a diagram showing probability traces of the protocol, which was proposed in [41] for card-based protocols. We believe that ball-based cryptography is a realization of usable security that helps people with understanding the principles of MPCs.

Outline

The remainder of this chapter is organized as follows. In Section 5.1, we propose ball-based cryptography by presenting a formal treatment of protocols using balls and bags. In Section 5.2, we show a pseudocode of the AND protocol with two inputs, and then show a diagram of the protocol, which implies correctness and security of the protocol. In Section 5.3, we extend the two-input AND protocol to an AND protocol with more than two inputs. In Section 5.4, we further extend the protocols to design general MPCs. In Section 5.5, we discuss the efficiency of our AND protocols. In Section 5.6, we show implementation examples for ball-based cryptography. We conclude this chapter in Section 5.7.

Related Work

Physical objects enable us to achieve cryptographic tasks, such as MPCs [15], zero-knowledge proofs [19], polling [50], and visual secret sharing [55]. As we perform these secure protocols with hand, their principles can be intuitively understood; hence, they are attractive. There are several researches on this subject (called real-life hands-on cryptography), such as a deck of playing cards [9, 11, 44] (known as *card-based cryptography*), tamper-evident seals [51], visual secret sharing sheets [10], coins [33], and a PEZ

[i] Although we believe that this automatic shuffle works well, shaking a bag after putting balls into it would be another way to achieve this.

dispenser [1, 4]. Real-life hands-on cryptography has been attracting many young people to the field of security and privacy.

Compared with their researches, our work employs the property of bags (namely, *automatic shuffle*) and utilize simple actions with colored balls and bags for performing MPCs. Related to probability theory, taking a ball out of a bag in ball-based cryptography can be a variant of an *urn problem* by regarding a bag as an urn. An urn problem often appears in probability theory and statistics, where a player takes one or more balls from an urn containing some balls. Some examples of urn problems such as binomial distribution are known. MPCs would be included in the collection of urn problems due to our work.

Comparison with Card-based Cryptography

Among related work introduced above, card-based cryptography is the most famous topic on real-life hands-on cryptography. We note that card-based cryptography and ball-based cryptography are different; a state of balls in a bag is denoted by a multiset (as will be seen in Section 5.1.1) while a playing card is defined as a fraction to represent two states of face-down and face-up [45].

Let us discuss the relation between them. Although it is true that a bag containing one ball can be regarded as a face-down card, balls and bags cannot be used to implement any card-based protocol. This is because, with balls and bags, it seems relatively difficult to realize a cyclic shuffle, which is used in a large number of card-based protocols to cyclically shuffle a sequence of cards. On the other hand, any ball-based protocol can be implemented by using a deck of playing cards because a bag containing balls can be represented by a set of face-down cards. That is, we can regard a pile of cards as a bag containing balls; by completely shuffling the pile of cards, the property of being disordered is guaranteed. For instance, the above two-party protocol can be implemented with two red cards and four white ones. However, such an implementation is no longer efficient because the famous five-card trick proposed by den Boer [11] requires only five cards; hence, they are different settings.

Overall, ball-based cryptography will open a new vista, and we expect that it will contribute to increasing people who are intersted in computer security and privacy.

5.1. Formalizing Protocols Based on Balls and Bags

In this section, we present a formal treatment of protocols based on balls and bags by constructing a model of ball-based secure computations.

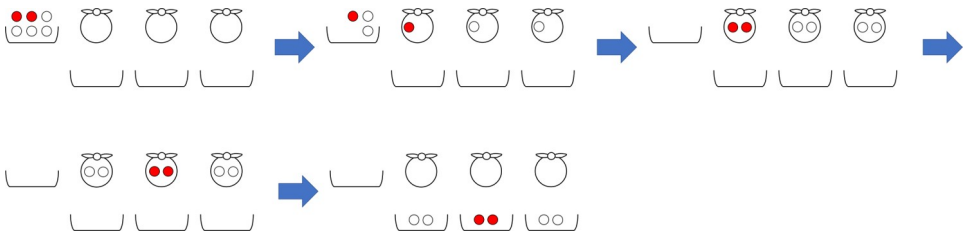


Figure 5.1: An example of executing our ball-based AND protocol with two inputs (when $a = b = 1$)

5.1.1. Notations

Remember the AND protocol (which computes $f(a, b) = a \wedge b$) proposed before. We now replace ingredients and saucepans with balls and bags, respectively. That is, we use two red balls, four white balls, and three bags as illustrated in Fig. 5.1; this is an example of executing our AND protocol using balls and bags when $a = b = 1$.

Let $\bullet$ and $\circ$ denote a red ball and a white ball, respectively. Assume that all balls have the same size. Seeing Fig. 5.1, notice that we have to consider two kinds of multisets of balls, namely a "bag" and a "tray." Thus, we introduce two expressions of a multiset of balls: $\{\{\cdot\}\}$ is *an invisible multiset* representing a bag possibly containing balls, and $[\cdot]$ is *a visible multiset* representing a tray. We use "invisible multiset" and "bag" interchangeably; we call an invisible multiset of balls $\{\{\mathsf{b}_1, \ldots, \mathsf{b}_\ell\}\}$ a *bag* (into which balls are put) where $\mathsf{b}_1, \ldots, \mathsf{b}_\ell \in \{\bullet, \circ\}$ for a natural number ℓ. Similarly, we call an visible multiset of balls $[\mathsf{b}_1, \ldots, \mathsf{b}_\ell]$ a *tray* (on which balls are put).

We assume that once balls are put into a bag, the order of the balls automatically becomes disordered. We call this property *automatically shuffled*. For example, $\{\{\bullet, \circ, \bullet\}\}$ and $\{\{\bullet, \bullet, \circ\}\}$ are indistinguishable to players. In this chapter, we write red balls first in an (in)visible multiset to unify the notation. It is interesting that constructing MPCs is possible even with balls and bags having such a property.

Using these notations, at the beginning of the two-party protocol, we have three empty bags (invisible multisets) $\{\{\}\}$, $\{\{\}\}$, $\{\{\}\}$ and a tray (visible multiset) $[\bullet, \bullet, \circ, \circ, \circ, \circ]$ on a table. Let us describe it with a tuple:

$$([\bullet, \bullet, \circ, \circ, \circ, \circ]; \{\{\}\}, \{\{\}\}, \{\{\}\}; [], [], []) \,. \tag{5.1}$$

We call this tuple a *configuration*. The first part of the configuration (before the first semi-colon) is a multiset of available balls, its second part (between the first and second semi-colons) is a sequence of bags, and its last part is a sequence of trays keeping balls picked from the bags. The last part may be omitted if there is no visible ball at that time.

Let us review the two-player protocol with these notations. At Step 2, Alice holds one red ball and two white balls, i.e., $[\bullet, \circ, \circ]$, and puts them

into the three bags depending on the value of a; the above configuration is transformed into as follows:

$$([\bullet,\bullet,\circ,\circ,\circ,\circ];\{\{\}\},\{\{\}\},\{\{\}\};[],[],[]) \to \begin{cases} ([\bullet,\circ,\circ];\{\{\circ\}\},\{\{\bullet\}\},\{\{\circ\}\};[],[],[]), & \text{if } a=0, \\ ([\bullet,\circ,\circ];\{\{\bullet\}\},\{\{\circ\}\},\{\{\circ\}\};[],[],[]), & \text{if } a=1. \end{cases} \tag{5.2}$$

At Step 3, Bob also puts balls depending on b; there are four possibilities:

$$\begin{aligned} &([];\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\};[],[],[]), && \text{if } (a,b)=(0,0), \\ &([];\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\};[],[],[]), && \text{if } (a,b)=(0,1), \\ &([];\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\};[],[],[]), && \text{if } (a,b)=(1,0), \\ &([];\{\{\bullet,\bullet\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[],[],[]), && \text{if } (a,b)=(1,1). \end{aligned} \tag{5.3}$$

Let p_{ij} for $i,j \in \{0,1\}$ represent the probability that the input (a,b) is (i,j). Then, the first line in Eq. (5.3) occurs with a probability of p_{00}, the second line occurs with a probability of p_{01}, and so on. Therefore, we now denote the above status Eq. (5.3) by

$$\begin{aligned} &([];\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\};[],[],[]) && (p_{00},0,0,0), \\ &([];\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\};[],[],[]) && (0,p_{01},0,0), \\ &([];\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\};[],[],[]) && (0,0,p_{10},0), \\ &([];\{\{\bullet,\bullet\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[],[],[]) && (0,0,0,p_{11}), \end{aligned} \tag{5.4}$$

where a 4-tuple $(q_{00},q_{01},q_{10},q_{11})$ on the right side means that q_{ij} is the conditional probability that $(a,b)=(i,j)$ and the current configuration is the left one.

Next, at Step 4, the three bags are shuffled; the resulting configurations will be:

$$\begin{aligned} &([];\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\};[],[],[]) && (\tfrac{p_{00}}{3},\tfrac{p_{01}}{3},\tfrac{p_{10}}{3},0), \\ &([];\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\};[],[],[]) && (\tfrac{p_{00}}{3},\tfrac{p_{01}}{3},\tfrac{p_{10}}{3},0), \\ &([];\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\};[],[],[]) && (\tfrac{p_{00}}{3},\tfrac{p_{01}}{3},\tfrac{p_{10}}{3},0), \\ &([];\{\{\bullet,\bullet\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[],[],[]) && (0,0,0,\tfrac{p_{11}}{3}), \\ &([];\{\{\circ,\circ\}\},\{\{\bullet,\bullet\}\},\{\{\circ,\circ\}\};[],[],[]) && (0,0,0,\tfrac{p_{11}}{3}), \\ &([];\{\{\circ,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\bullet\}\};[],[],[]) && (0,0,0,\tfrac{p_{11}}{3}). \end{aligned} \tag{5.5}$$

Finally, Alice and Bob take all the balls out of the three bags; this is denoted by the last part of the configurations, namely visible multisets, as

follows:

$$\begin{array}{ll}
([],\{\{\}\},\{\{\}\},\{\{\}\};[\circ,\circ],[\bullet,\circ],[\bullet,\circ]) & \left(\frac{p_{00}}{p_0},\frac{p_{01}}{p_0},\frac{p_{10}}{p_0},0\right), \\
([],\{\{\}\},\{\{\}\},\{\{\}\};[\bullet,\circ],[\bullet,\circ],[\circ,\circ]) & \left(\frac{p_{00}}{p_0},\frac{p_{01}}{p_0},\frac{p_{10}}{p_0},0\right), \\
([],\{\{\}\},\{\{\}\},\{\{\}\};[\bullet,\circ],[\circ,\circ],[\bullet,\circ]) & \left(\frac{p_{00}}{p_0},\frac{p_{01}}{p_0},\frac{p_{10}}{p_0},0\right), \\
([],\{\{\}\},\{\{\}\},\{\{\}\};[\bullet,\bullet],[\circ,\circ],[\circ,\circ]) & (0,0,0,1), \\
([],\{\{\}\},\{\{\}\},\{\{\}\};[\circ,\circ],[\bullet,\bullet],[\circ,\circ]) & (0,0,0,1), \\
([],\{\{\}\},\{\{\}\},\{\{\}\};[\circ,\circ],[\circ,\circ],[\bullet,\bullet]) & (0,0,0,1),
\end{array} \tag{5.6}$$

where $p_0 = p_{00} + p_{01} + p_{10}$. The last part of the configurations, i.e., balls picked from each bag, tells us that the AND value $a \wedge b$ is obtained depending on whether two red balls are taken from the same bag or not and that no information except for the output is leaked (see Section 5.2 for more rigorous discussion).

5.1.2. Definition of Protocols

Let us introduce a formal definition of protocols based on balls and bags.

Before going into the details, we first introduce extended operations for a sequence of multisets. Let $\mathbf{X} = (X_1, \ldots, X_k)$ and $\mathbf{Y} = (Y_1, \ldots, Y_k)$ be sequences of multisets of the same length k. We define the following two operations: $\mathbf{X} \cup \mathbf{Y} \coloneqq (X_1 \cup Y_1, \ldots, X_k \cup Y_k)$ and $\mathsf{union}(\mathbf{X}) \coloneqq X_1 \cup \cdots \cup X_k$.

Next, we formally define a *configuration* (examples of which were seen in Section 5.1.1).

Definition 1 (Configuration) *Let D be a multiset of balls and $k \geq 1$ be an integer (representing the number of bags). We call a triple $(T_0; \mathbf{B}; \mathbf{T}) = (T_0; B_1, \ldots, B_k; T_1, \ldots, T_k)$ a* configuration *if it satisfies the following:*

- *$T_0 \subseteq D$ is a tray, where all balls in D are put here before the execution of a protocol;*
- *$\mathbf{B} = (B_1, \ldots, B_k) \subseteq D^k$ is a sequence of k bags;*
- *$\mathbf{T} = (T_1, \ldots, T_k) \subseteq D^k$ is a sequence of k trays representing that balls in T_i were taken out of B_i for every i, $1 \leq i \leq k$;*
- *$T_0 \cup \mathsf{union}(\mathbf{B} \cup \mathbf{T}) = D$.*

We denote by $\mathcal{C}^{(D,k)}$ the set of all configurations derived by fixing D and k.

Given a configuration $(T_0; B_1, \ldots, B_k; T_1, \ldots, T_k)$, balls in trays T_0 and $T_1, \ldots, T_k$ are visible while balls in bags $B_1, \ldots, B_k$ are invisible. We assume that the number of balls inside each bag is known to the public. Bearing this in mind, we define a *visible configuration* $\mathsf{vis}(C)$ for a configuration $C =$

$(T_0; B_1, \ldots, B_k; T_1, \ldots, T_k)$ as follows: $\mathsf{vis}(C) \coloneqq (T_0; |B_1|, \ldots, |B_k|; T_1, \ldots, T_k)$, where $|B_i|$ denotes the number of elements (balls) in B_i. We also define the set of all visible configurations as $\mathsf{Vis}^{(D,k)} = \left\{\mathsf{vis}(C) \mid C \in \mathcal{C}^{(D,k)}\right\}$.[ii]

We are now ready to formally define a "protocol" $\mathcal{P}$ achieving MPCs using balls and bags.

Definition 2 (Protocol) *A* protocol $\mathcal{P}$ *is a tuple* (D, k, n, U, Q, A) *satisfying the following:*

- D *is a multiset over* $\{\bullet, \circ\}$*, representing balls used in the protocol and* $k \geq 1$ *is the number of bags; therefore, the initial configuration is* $C^0 = (D; \mathbf{B}^0; \mathbf{T}^0)$ *such that* $\mathsf{union}(\mathbf{B}^0 \cup \mathbf{T}^0) = \phi$ *and* $|\mathbf{B}^0| = |\mathbf{T}^0| = k$*.*
- $n \geq 2$ *represents the number of players participating in the protocol.*
- U *is the set of players' possible inputs. In the sequel, we fix it to* $U = \{0,1\}^n$*, meaning that each player's input is a bit.*
- Q *is a set of states with two distinguished states, namely, the initial state* q_0 *and the final state* q_f*.*
- $A : (Q \setminus \{q_f\}) \times \mathsf{Vis}^{(D,k)} \to Q \times \mathsf{Action}$ *is an action function, which specifies the next state and an action, given a current state and a visible configuration. The set* Action *includes the following actions, where we describe each action for a configuration* $C = (T_0; \mathbf{B}; \mathbf{T}) = (T_0; B_1, \ldots, B_k; T_1, \ldots, T_k)$*.*
  - $(\mathsf{PublicPut}, \mathsf{b}, p)$ *for* $\mathsf{b} \in T_0$ *and* $p \in \{1, 2, \ldots, k\}$*: This puts the ball* b *from the tray* T_0 *into the* p*-th bag* B_p *publicly (i.e., the color of the ball is known to all players). That is, it transforms* C *into the following configuration* C'*:*

    $$C' = (T_0 \setminus [\mathsf{b}]; B_1, \ldots, B_{p-1}, B_p \cup \{\{\mathsf{b}\}\}, B_{p+1}, \ldots, B_k; \mathbf{T}).$$

    Note that the player executing this action must show the ball b *to other players before putting it into the bag.*
  - $(\mathsf{PrivatePut}, i, \mathbf{I}_0, \mathbf{I}_1)$ *for* i*,* $1 \leq i \leq n$*, and sequences of* k *multisets* $\mathbf{I}_0 = (I_0^1, \ldots, I_0^k)$ *and* $\mathbf{I}_1 = (I_1^1, \ldots, I_1^k)$ *such that* $\mathsf{union}(\mathbf{I}_0) = \mathsf{union}(\mathbf{I}_1) \subseteq T_0$ *and* $|I_0^j| = |I_1^j|$ *for every* j*,* $1 \leq j \leq k$*: This makes the* i*-th player holding an input* $x_i \in \{0,1\}$ *take balls from* T_0 *and then privately put them into* $\mathbf{B}$ *as specified by* $\mathbf{I}_{x_i}$*. That is, it transforms* C *into the following configuration* C'*:*

    $$C' = (T_0 \setminus \mathsf{union}(\mathbf{I}_{x_i}); \mathbf{B} \cup \mathbf{I}_{x_i}; \mathbf{T}).$$

    Because $\mathsf{union}(\mathbf{I}_0) = \mathsf{union}(\mathbf{I}_1)$*, the numbers of* $\bullet$ *and* $\circ$ *in* $\mathbf{I}_0$ *and* $\mathbf{I}_1$ *are the same.*

[ii] Information about a configuration transition (e.g., a red ball was moved to the second bag) can be captured by a visible "configuration-trace" that will be mentioned later.

– (Shuf, R) *for* $R \subseteq \{1, 2, \ldots, k\}$*: This shuffles bags specified by* R *so that the resulting order of the bags becomes unknown to all players. That is, it transforms* C *into the following configuration* C'*:*

$$C' = \left(T_0; B_{\pi^{-1}(1)}, B_{\pi^{-1}(2)}, \ldots, B_{\pi^{-1}(k)}; \mathbf{T}\right),$$

where π *is uniformly drawn at random from the set of all permutations such that all positions except for* R *are fixed points (i.e.,* $\pi(i) = i$ *for any* $i \notin R$*).*

– (Take, p) *for* $p \in \{1, 2, \ldots, k\}$*: This takes a ball out of the* p*-th bag* B_p *and then the ball is put on the tray* T_p*. That is, it transforms* C *into the following configuration* C'*:*

$$\begin{aligned} &C' \coloneqq (T_0; \mathbf{B}'; \mathbf{T}'), \text{ where} \\ &\mathbf{B}' = (B_1, \ldots, B_{p-1}, B_p \setminus \{\{\mathsf{b}\}\}, B_{p+1}, \ldots, B_k), \text{ and} \\ &\mathbf{T}' = (T_1, \ldots, T_{p-1}, T_p \cup [\mathsf{b}], T_{p+1}, \ldots, T_k), \end{aligned}$$

for a (taken) ball $\mathsf{b} \in B_p$*. Note that the taken ball was drawn uniformly at random from* B_p*. Also note that, during this action, no player can get no information about other balls in* B_p*. If we take all balls in all bags, we write this action as* $(\mathsf{TakeAll})$*.*

– $(\mathsf{Move}, \mathsf{b}, p)$ *for* $\mathsf{b} \in T_p$ *and* $p \in \{1, 2, \ldots, k\}$*: This moves the ball* b *on the* p*-th tray* T_p *to the tray* T_0*. That is, it transforms* C *into the following configuration* C'*:*

$$C' = (T_0 \cup [\mathsf{b}]; \mathbf{B}; T_1, \ldots, T_{p-1}, T_p \setminus [\mathsf{b}], T_{p+1}, \ldots, T_k).$$

If we move all balls on all trays, we write this action as $(\mathsf{MoveAll})$*.*

– $(\mathsf{MergeBags}, p_1, p_2)$ *for* $p_1, p_2 \in \{1, 2, \ldots, k\}$*: This merges the* p_1*-th bag with the* p_2*-th bag, i.e., all balls in* B_{p_1} *are moved into* B_{p_2} *without revealing the colors of the balls. That is, it transforms* C *into the following configuration* C'*:*

$$C' = (T_0; B_1, \ldots, B_{p_1-1}, \{\{\}\}, \ldots, B_{p_2} \cup B_{p_1}, \ldots, B_k; \mathbf{T}).$$

– (Return, e) *for some expression* e*. This special action indicates that the protocol terminates with the output* e*.*

A protocol $\mathcal{P} = (D, k, n, U, Q, A)$ proceeds as follows, given inputs $x = (x_1, \ldots, x_n) \in U = \{0, 1\}^n$. Balls and bags corresponding to D and k, respectively, are on the table, i.e., the initial configuration is $C^0 = (D; \mathbf{B}^0; \mathbf{T}^0)$ where $\mathsf{union}(\mathbf{B}^0 \cup \mathbf{T}^0) = \phi$ and $|\mathbf{B}^0| = |\mathbf{T}^0| = k$. Each i-th player privately holds an input $x_i \in \{0, 1\}$, and the state of the protocol is $q_0 \in Q$. Then, the configuration and the state are transformed according to the output of the action function $A(q_0, \mathsf{vis}(C_0))$. The protocol continues to apply the action

function A with its current configuration and state being transformed until the state becomes q_f; it terminates with (Return, e).

Let f be a Boolean function such that $f : \{0,1\}^n \to \{0,1\}$. We say that a protocol $\mathcal{P}$ is *correct* for f if e (which is an output of Return) derived by executing $\mathcal{P}$ is always equivalent to the value of $f(x_1, \ldots, x_n)$.

Next, let us consider *security* of a protocol $\mathcal{P}$. We mention some terms. Consider an execution of a protocol $\mathcal{P}$; the enumeration of all visible configurations $(\mathsf{vis}\,(C_0)\,, \mathsf{vis}\,(C_1)\,, \ldots, \mathsf{vis}\,(C_t))$ from the initial to the final one (where C_{i-1} is transformed into C_i by an action) is called a *visible configuration-trace* (of $\mathcal{P}$).

Definition 3 (Security) *Let $\mathcal{P} = (D, k, n, U, Q, A)$ be a protocol which is correct for f. Let V be the random variable representing the visible configuration-trace of $\mathcal{P}$, M be the random variables representing the inputs of $\mathcal{P}$, and F be the random variable representing the output of f. We say that $\mathcal{P}$ is* secure *for f if it satisfies*

$$\Pr[M = x \mid F = 0] = \Pr[M = x \mid V = v, F = 0]\,, \text{ and}$$
$$\Pr[M = x \mid F = 1] = \Pr[M = x \mid V = v, F = 1]\,,$$

for any $x \in U$ and visible configuration-trace v.

The security of $\mathcal{P}$ intuitively means that no information except for the output is leaked from balls taken out of bags.

5.2. AND Protocol with Two Inputs

In this section, based on the definitions in Section 5.1.2, we present a formal description of our AND protocol with two inputs $(x_1, x_2) \in \{0,1\}^2$ introduced in Section 5.1.1. We first review the principle of our AND protocol and then present its description. Finally, a diagram of the AND protocol is given in Fig. 5.2, from which its correctness and security can be confirmed.

5.2.1. Principle and Description

Remember the AND protocol introduced in Section 5.1.1. Alice and Bob put $\bullet$ into the (first) bag B_1 if his/her private bit is 1; otherwise, they are supposed to put $\bullet$ into different bags, B_2 and B_3.

More formally, Alice acts by $\Big(\mathsf{PrivatePut}, 1, ([\circ], [\bullet], [\circ]), ([\bullet], [\circ], [\circ])\Big)$, and the possible configurations will be as in Eq. (5.2). Then, Bob holding x_2 acts by $\Big(\mathsf{PrivatePut}, 2, ([\circ], [\circ], [\bullet]), ([\bullet], [\circ], [\circ])\Big)$, and the possible configurations will be as in Eq. (5.4). After shuffling the three bags and then taking all balls, they can know that the AND value is 1 if there is $[\bullet, \bullet]$; otherwise, 0. No information about the inputs (beyond the output) is leaked.

Protocol 3. The two-input AND protocol:
$([\bullet, \bullet, \circ, \circ, \circ, \circ], 3, 2, \{0,1\}^2, Q, A)$.

1. $(\mathsf{PrivatePut}, 1, ([\circ], [\bullet], [\circ]), ([\bullet], [\circ], [\circ]))$
2. $(\mathsf{PrivatePut}, 2, ([\circ], [\circ], [\bullet]), ([\bullet], [\circ], [\circ]))$
3. $(\mathsf{Shuf}, \{1, 2, 3\})$
4. $(\mathsf{TakeAll})$
5. **if** visible conf. includes $[\bullet, \bullet]$ **then**
6. $\quad(\mathsf{Return}, \text{“}x_1 \wedge x_2 = 1\text{”})$
7. **else**
8. $\quad(\mathsf{Return}, \text{“}x_1 \wedge x_2 = 0\text{”})$

As seen above, our two-input AND protocol uses six balls and three bags; a formal description of the protocol is shown in Protocol 3. See Table 5.1 for the performance of the protocol and Section 5.5 is devoted to the discussion about it (as well as that of our multi-input AND protocol which will be presented in Section 5.3).

5.2.2. Security: A diagram of status transitions

To confirm the correctness and security of the AND protocol, we construct an diagram in Fig. 5.2 showing status transitions of the protocol. This method was first proposed in [31] for card-based protocols, and then an extended diagram was proposed in [41], which uses the *probability trace* below.

Definition 4 (Probability Trace) *Let $\mathcal{P}$ be a protocol with an input set $U = \{0,1\}^n$, and let v be a visible configuration-trace. We regard every input $x \in U = \{0,1\}^n$ as a decimal number x, $1 \leq x \leq |U| = 2^n$. A $|U|$-tuple $(q_1, q_2, \ldots, q_{|U|})$ such that $q_x = \Pr[M = x, G_v = C \mid V = v]$ for every $x \in U$ is called a* probability trace *for a configuration C and the visible configuration-trace v, where M, G_v, and V are random variables of the original input, configuration when v is seen, and visible configuration-trace, respectively.*

See Fig. 5.2 again. Each “box” in the figure including several pairs of a configuration and a probability trace is called a *status*. Each status is associated with a prefix of the visible configuration-trace. As stated in Section 5.1.1 and formally defined in Definition 4, a probability trace next to a configuration represents the conditional probability that the current configuration is the configuration, given the prefix of the visible sequence-trace.

A status in Fig. 5.2 is transformed into the next status by an action as follows:

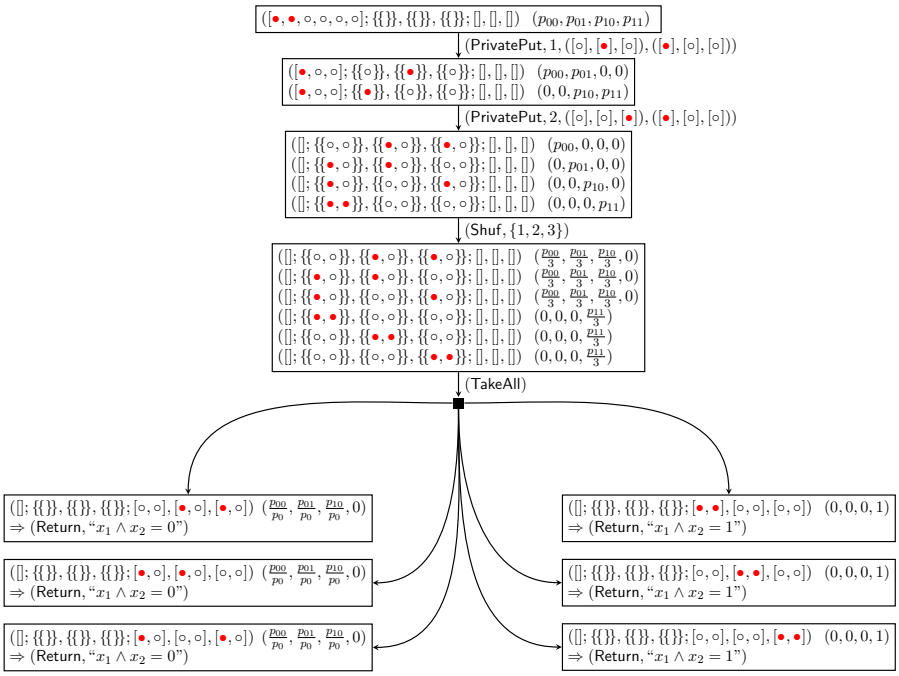


Figure 5.2: A diagram of the two-input AND protocol with two inputs $x_1, x_2 \in \{0, 1\}$ introduced in Section 5.1.1, where $p_0 = p_{00} + p_{01} + p_{10}$.

- The topmost status consists of a single pair of the initial configuration C^0 and the probability trace $(p_{00}, p_{01}, p_{10}, p_{11})$.
- The first (and second) action is PrivatePut. For instance, when $x_1 = 0$, three balls specified by $\mathbf{I}_0^1 = ([\circ], [\bullet], [\circ])$ are privately put into the bags, and the probability trace becomes $(p_{00}, p_{01}, 0, 0)$.
- The third action is Shuf. After the action, there are three possible configurations with the equal probability, i.e., 1/3. This can be seen in the coefficient in the probability traces for the three configurations.
- The fourth action is TakeAll, yielding six visible configurations. Assume for example that we have observed $([\circ, \circ], [\bullet, \circ], [\bullet, \circ])$. The fourth coordinate in the probability trace is 0, and other coordinates are all $\frac{p_{00}+p_{01}+p_{10}}{3}$, and hence, we know that the inputs must not be (1,1). It means that the output is "$x_1 \wedge x_2 = 0$". Furthermore, no information about the inputs is leaked because the distribution of the conditional probability that the inputs are (0,0), (0,1), and (1,0) is the same as the one of knowing $x_1 \wedge x_2$ before the execution of the protocol.

5.3. AND Protocol with More Than Two Inputs

In this section, we deal with secure AND computation with more than two inputs. That is, we present a general AND protocol that securely computes $x_1 \wedge \cdots \wedge x_n$, given that n players $P_1, \ldots, P_n$ hold private input bits $x_1, \ldots, x_n \in \{0, 1\}$, respectively.

5.3.1. Idea and Description

Simply extending the two-input AND does not work. As stated in Section 5.2.1, our two-input AND protocol computes the AND value by making two players privately put $\bullet$ into the same bag if and only if their private bits both are 1. If we simply extend this principle, can we construct an n-input AND protocol for any n? Consider for instance that there are three players P_1, P_2, and P_3 where P_i holds his/her private bit $x_i \in \{0, 1\}$ for every i, $1 \leq i \leq 3$. We prepare four (empty) bags B_1, B_2, B_3, and B_4 and make each player P_i privately put $\bullet$ into the (first) bag B_1 if x_i is 1 (otherwise, into the $(i{+}1)$-st bag B_{i+1}) and $\circ$ into the remaining bags. Then, the resulting configurations will be as follows:

$$\begin{array}{ll}
(;\{\{\circ,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\};) & (p_{000},0,0,0,0,0,0,0),\\
(;\{\{\bullet,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\};) & (0,p_{001},0,0,0,0,0,0),\\
(;\{\{\bullet,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\};) & (0,0,p_{010},0,0,0,0,0),\\
(;\{\{\bullet,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\};) & (0,0,0,p_{100},0,0,0,0),\\
(;\{\{\bullet,\bullet,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\};) & (0,0,0,0,p_{011},0,0,0),\\
(;\{\{\bullet,\bullet,\circ\}\},\{\{\circ,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\};) & (0,0,0,0,0,p_{101},0,0),\\
(;\{\{\bullet,\bullet,\circ\}\},\{\{\circ,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\},\{\{\bullet,\circ,\circ\}\};) & (0,0,0,0,0,0,p_{110},0),\\
(;\{\{\bullet,\bullet,\bullet\}\},\{\{\circ,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\},\{\{\circ,\circ,\circ\}\};) & (0,0,0,0,0,0,0,p_{111}),
\end{array} \tag{5.7}$$

where we omit the tray T_0 and empty trays. Therefore, after shuffling the four bags and taking all balls, all the players can know that the AND value is 1 if there is $[\bullet, \bullet, \bullet]$; otherwise, 0. As seen from Eq. (5.7), however, they obtain additional information about the inputs from the number of $\bullet$ in a tray; for example, if there is $[\bullet, \bullet, \circ]$, it means that the number of 1 among the inputs x_1, x_2, and x_3 is two. Therefore, this straightforward extension is not a secure computation of AND.

Our idea Let us go back to Step 4 in the AND protocol introduced in Section 5.1.1 (or the fourth status in Fig. 5.2). Suppose that we replace the action (TakeAll) with (Take, 1), i.e., taking a ball out of the (first) bag B_1 in Eq. (5.5). There are two possibilities, i.e., $\bullet$ is taken with a probability of 1/3 or $\circ$ is taken with a probability of 2/3.[iii] If it is $\bullet$, the configurations in

[iii] Remember that in Step 3, the three bags have been shuffled. Hence, taking a ball out of B_1 does not leak information about x_1 and x_2.

Eq. (5.5) are transformed into the followings:[iv]

$$\begin{array}{ll}([];\{\{\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\};[\bullet],[],[]) & (\frac{p_{00}}{2},\frac{p_{01}}{2},\frac{p_{10}}{2},0),\\ ([];\{\{\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\};[\bullet],[],[]) & (\frac{p_{00}}{2},\frac{p_{01}}{2},\frac{p_{10}}{2},0),\\ ([];\{\{\bullet\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[\bullet],[],[]) & (0,0,0,p_{11}).\end{array} \tag{5.8}$$

Note that this $(\mathsf{Take},1)$ action does not leak any information about the inputs because the coordinate-wise sum of the probability traces in Eq. (5.8) is equal to $(p_{00},p_{01},p_{10},p_{11})$, meaning that the (conditional) distribution on inputs does not change. Then, consider that we add the fourth bag and make the third player P_3 act by

$$\Big(\mathsf{PrivatePut},3,x_3,([\circ],[],[],[\bullet,\circ]),([\bullet],[],[],[\circ,\circ])\Big).$$

That is, if $x_3=0$, the configurations in Eq. (5.8) are transformed into

$$\begin{array}{ll}([];\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\};[\bullet],) & (\frac{p_{000}}{2},0,\frac{p_{010}}{2},\frac{p_{100}}{2},0,0,0,0),\\ ([];\{\{\circ,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\};[\bullet],) & (\frac{p_{000}}{2},0,\frac{p_{010}}{2},\frac{p_{100}}{2},0,0,0,0),\\ ([];\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\};[\bullet],) & (0,0,0,0,0,0,p_{110},0).\end{array}$$

If $x_3=1$, the configurations in Eq. (5.8) are transformed into

$$\begin{array}{ll}([];\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[\bullet],) & (0,\frac{p_{001}}{2},0,0,\frac{p_{011}}{2},\frac{p_{101}}{2},0,0),\\ ([];\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\};[\bullet],) & (0,\frac{p_{001}}{2},0,0,\frac{p_{011}}{2},\frac{p_{101}}{2},0,0),\\ ([];\{\{\bullet,\bullet\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[\bullet],) & (0,0,0,0,0,0,0,p_{111}).\end{array}$$

As seen from the above configurations, there is $\{\{\bullet,\bullet\}\}$ if the AND value $x_1\wedge x_2\wedge x_3$ is 1; otherwise, the four bags are a permuted sequence of $\{\{\bullet,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\}$. Thus, after $(\mathsf{Shuf},\{1,2,3,4\})$ and $(\mathsf{TakeAll})$, all the players can obtain the AND value of the three inputs $x_1\wedge x_2\wedge x_3$ without revealing any information except for the output.

Next, let us consider $n=4$. In the same way as $n=3$, assume that we replace the above action $(\mathsf{TakeAll})$ (in the three-input protocol) with $(\mathsf{Take},1)$ and then a red ball is taken. The resulting configurations will be the followings:

$$\begin{array}{ll}(;\{\{\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[\bullet],) & (\frac{p_{000}}{3},\frac{p_{001}}{3},\frac{p_{010}}{3},\frac{p_{100}}{3},\frac{p_{011}}{3},\frac{p_{101}}{3},\frac{p_{110}}{3},0),\\ (;\{\{\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\},\{\{\circ,\circ\}\};[\bullet],) & (\frac{p_{000}}{3},\frac{p_{001}}{3},\frac{p_{010}}{3},\frac{p_{100}}{3},\frac{p_{011}}{3},\frac{p_{101}}{3},\frac{p_{110}}{3},0),\\ (;\{\{\circ\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\},\{\{\bullet,\circ\}\};[\bullet],) & (\frac{p_{000}}{3},\frac{p_{001}}{3},\frac{p_{010}}{3},\frac{p_{100}}{3},\frac{p_{011}}{3},\frac{p_{101}}{3},\frac{p_{110}}{3},0),\\ (;\{\{\bullet\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\},\{\{\circ,\circ\}\};[\bullet],) & (0,0,0,0,0,0,0,p_{111}).\end{array} \tag{5.9}$$

These are the "same" configurations as in Eq. (5.8), i.e., the first bag contains a red ball if and only if $x_1=x_2=x_3=1$. Thus, a four-input AND protocol can be constructed by making P_4 privately put balls in a similar way to P_3 when $n=3$.

In this way, we can extend the two-input AND protocol to the n-input one for $n\geq 3$.

[iv] If it is $\circ$, we return $\circ$ into B_1 and then repeat shuffling bags and taking a ball out of B_1 until $\bullet$ is taken.

Protocol 4. The AND protocol with $n\,(\geq 3)$ inputs:
$([\bullet, \bullet, \circ, \circ, \ldots, \circ], n+1, n, \{0,1\}^n, Q, A)$.

1. $(\mathsf{PrivatePut}, 1, \mathbf{I}_0^1, \mathbf{I}_1^1)$
2. $(\mathsf{PrivatePut}, 2, \mathbf{I}_0^2, \mathbf{I}_1^2)$
3. **for** $i \leftarrow 3$ to $n-1$ **do**
4. **while**(1)
5. $(\mathsf{Shuf}, \{1, 2, \ldots, i\})$
6. $(\mathsf{Take}, 1)$
7. **if** taken ball $= \bullet$ **then**
8. $(\mathsf{Move}, \bullet, 1)$
9. $(\mathsf{PrivatePut}, i, \mathbf{I}_0^i, \mathbf{I}_1^i)$
10. **break**
11. **else if** taken ball $= \circ$ **then**
12. $(\mathsf{Move}, \circ, 1)$
13. $(\mathsf{PublicPut}, \circ, 1)$
14. $(\mathsf{PrivatePut}, n, \mathbf{I}_0^n, \mathbf{I}_1^n)$
15. $(\mathsf{Shuf}, \{1, 2, \ldots, n+1\})$
16. $(\mathsf{TakeAll})$
17. **if** visible conf. includes $[\bullet, \bullet]$ **then**
18. $(\mathsf{Return}, \text{“}x_1 \wedge \cdots \wedge x_n = 1\text{”})$
19. **else**
20. $(\mathsf{Return}, \text{“}x_1 \wedge \cdots \wedge x_n = 0\text{”})$

Description. Based on the idea explained above, we present a formal description of our AND protocol with $n\,(\geq 3)$ inputs in Protocol 4. Here,

$$\mathbf{I}_0^1 = \Big([\circ], [\bullet], [\circ], [], \ldots, []\Big),$$
$$\mathbf{I}_0^2 = \Big([\circ], [\circ], [\bullet], [], \ldots, []\Big),$$
$$\mathbf{I}_1^1 = \mathbf{I}_1^2 = \Big([\bullet], [\circ], [\circ], [], \ldots, []\Big),$$

and for every i, $3 \leq i \leq n$,

$$\mathbf{I}_0^i = \Big(\overset{1}{[\circ]}, \overset{2}{[]}, \ldots, \overset{i-1}{[]}, \overset{i}{[\bullet, \circ]}, \overset{i+1}{[]}, \ldots, \overset{k}{[]}\Big),$$
$$\mathbf{I}_1^i = \Big([\bullet], [], \ldots, [], [\circ, \circ], [], \ldots, []\Big).$$

5.3.2. Correctness and Security

Figure 5.3 shows a part of the diagram of the n-input AND protocol described in Section 5.3.1; the partial diagram corresponds to Steps 3–13 in Protocol 4. From this figure, we can see that there is $\{\{\bullet, \bullet\}\}$ if and only if $x_1 = x_2 = \cdots = x_i = 1$.

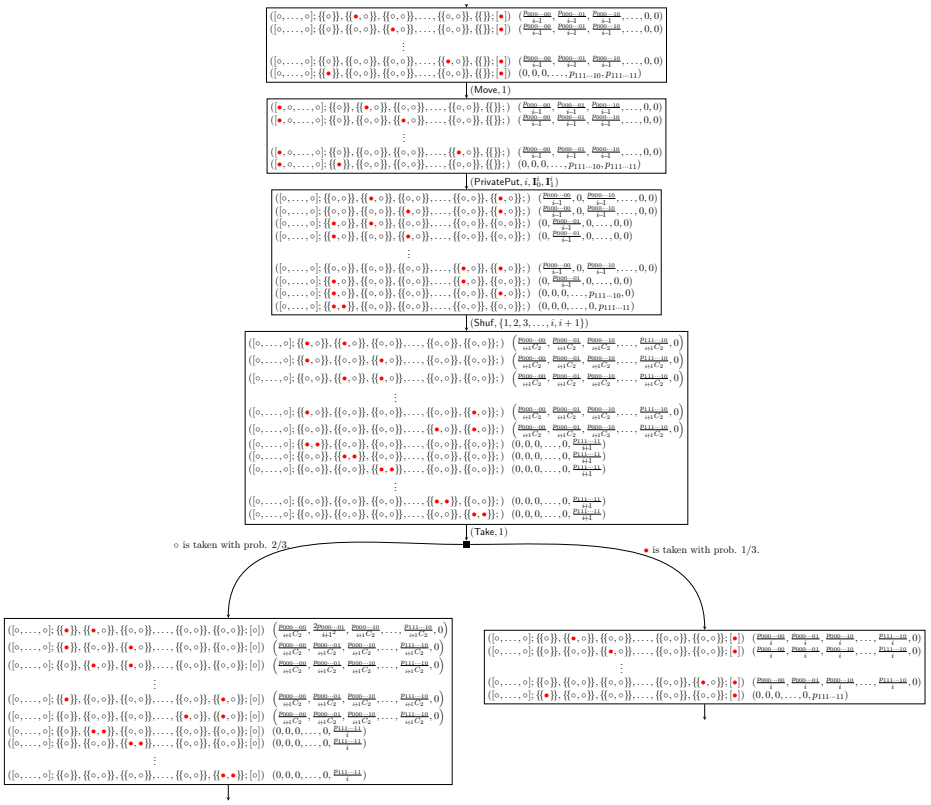


Figure 5.3: A crucial part of a diagram of the AND protocol with n inputs $x_1, x_2, \ldots, x_n \in \{0,1\}^n$ for the loop index $i \in \{3, 4, \ldots, n-1\}$, where empty trays and empty bags (over the $(i+2)$-nd) are omitted for simplicity.

We can furthermore see that $(\mathsf{Take}, 1)$ leaks no information about the inputs because the probability of taking a red ball is always $\frac{1}{i+1}$, which is independent of the inputs. Therefore, this protocol is secure.

5.4. Protocols for Any Boolean Function

In this section, we will explain how to realize protocols for any Boolean function with balls and bags. To achieve it, we will propose *committed-format* AND and NOT protocols that are known to be functionally complete. Note that in our AND protocols shown in Sections 5.2 and 5.3, the balls should be revealed to obtain the AND values in the final step. Namely, the previous AND protocols cannot be used as building blocks for a composition of another protocol. As a building block for a composite protocol, the following property called *committed-format* is required: We say a protocol is committed-format if its output can be an input for another (next) protocol without revealing balls themselves.

Protocol 5. The committed-format AND protocol, where the number of balls in each bag is denoted by m:
$([\circ, \ldots, \circ], 4, 2, \{0,1\}^2, Q, A)$.

Initial state:

$([\circ,\ldots,\circ]; \{\{\circ,\circ,\ldots,\circ\}\}, \{\{\bullet,\circ,\ldots,\circ\}\}, \{\{\circ,\circ,\ldots,\circ\}\}, \{\{\bullet,\circ,\ldots,\circ\}\}; [], [], [], [])$ $(p_{00}, 0, 0, 0)$

$([\circ,\ldots,\circ]; \{\{\circ,\circ,\ldots,\circ\}\}, \{\{\bullet,\circ,\ldots,\circ\}\}, \{\{\bullet,\circ,\ldots,\circ\}\}, \{\{\circ,\circ,\ldots,\circ\}\}; [], [], [], [])$ $(0, p_{01}, 0, 0)$

$([\circ,\ldots,\circ]; \{\{\bullet,\circ,\ldots,\circ\}\}, \{\{\circ,\circ,\ldots,\circ\}\}, \{\{\circ,\circ,\ldots,\circ\}\}, \{\{\bullet,\circ,\ldots,\circ\}\}; [], [], [], [])$ $(0, 0, p_{10}, 0)$

$([\circ,\ldots,\circ]; \{\{\bullet,\circ,\ldots,\circ\}\}, \{\{\circ,\circ,\ldots,\circ\}\}, \{\{\bullet,\circ,\ldots,\circ\}\}, \{\{\circ,\circ,\ldots,\circ\}\}; [], [], [], [])$ $(0, 0, 0, p_{11})$

Steps:

1. $(\mathsf{MergeBags}, 3, 1)$
2. $(\mathsf{PublicPut}, \circ \cdots \circ, 2)$
3. $(\mathsf{PublicPut}, \circ \cdots \circ, 4)$
4. **while**(1)
5. $\quad(\mathsf{Shuf}, \{1,2,4\})$
6. $\quad(\mathsf{Take}, 1)$
7. $\quad$**if** taken ball $= \bullet$ **then**
8. $\quad\quad$**break**
9. $\quad$**else**
10. $\quad\quad(\mathsf{Move}, \circ, 1)$
11. $\quad\quad(\mathsf{PublicPut}, \circ, 1)$
12. $(\mathsf{MergeBags}, 4, 2)$
13. $(\mathsf{PublicPut}, \circ \cdots \circ, 1)$
14. $(\mathsf{Result}, 1, 2)$

Note that in the sequel, we will construct committed-format protocols with balls, where sizes of all balls are the same.

5.4.1. Committed-Format AND protocol

Encoding. We encode a Boolean value with two bags, each of which includes the same number of balls, as follows:

$$\begin{aligned} \{\{\circ,\circ,\circ,\ldots,\circ\}\}, \{\{\bullet,\circ,\circ,\ldots,\circ\}\} &= 0, \\ \{\{\bullet,\circ,\circ,\ldots,\circ\}\}, \{\{\circ,\circ,\circ,\ldots,\circ\}\} &= 1. \end{aligned} \tag{5.10}$$

That is, a pair of bags where the second (resp. first) bag contains exactly one red ball $\bullet$ represents 0 (resp. 1). A pair of bags representing a bit $x \in \{0,1\}$ according to the above encoding rule is called a *commitment to* x.

Let us replace a Return action in Definition 2 with a Result action for a committed-format protocol: $(\mathsf{Result}, p_1, p_2)$ for $p_1, p_2 \in \{1, \ldots, k\}$. This means that the protocol terminates with the commitment consisting of the p_1-th and p_2-th bags.

Idea. Given two commitments to $x_1 \in \{0,1\}$ and $x_2 \in \{0,1\}$, a committed-format AND protocol produces a commitment to $x_1 \wedge x_2$. Let B_1 and B_2 be two bags constituting a commitment to x_1, and B_3 and B_4 be those constituting a commitment to x_2 such that $|B_1| = |B_2| = |B_3| = |B_4| = m$. Suppose that we merge B_3 with B_1 via $\mathsf{MergeBags}$ (*i.e.*, B_1 and B_3 become such that $|B_1| = 2m$ and $|B_3| = 0$) and put m white balls into each of B_2 and B_4 (*i.e.*, they become such that $|B_2| = 2m$ and $|B_4| = 2m$). Then, the resulting configurations will be as follows:

$$\begin{aligned}
&(;\{\{\circ,\circ,\circ,\dots,\circ\}\},\{\{\bullet,\circ,\circ,\dots,\circ\}\},\{\{\}\},\{\{\bullet,\circ,\circ,\dots,\circ\}\};) && (p_{00},0,0,0),\\
&(;\{\{\bullet,\circ,\circ,\dots,\circ\}\},\{\{\bullet,\circ,\circ,\dots,\circ\}\},\{\{\}\},\{\{\circ,\circ,\circ,\dots,\circ\}\};) && (0,p_{01},0,0),\\
&(;\{\{\bullet,\circ,\circ,\dots,\circ\}\},\{\{\circ,\circ,\circ,\dots,\circ\}\},\{\{\}\},\{\{\bullet,\circ,\circ,\dots,\circ\}\};) && (0,0,p_{10},0),\\
&(;\{\{\bullet,\bullet,\circ,\dots,\circ\}\},\{\{\circ,\circ,\circ,\dots,\circ\}\},\{\{\}\},\{\{\circ,\circ,\circ,\dots,\circ\}\};) && (0,0,0,p_{11}),
\end{aligned}$$

where we omit the tray T_0. Note that these are similar to Eq. (5.4) if we eliminate the empty bag B_3, i.e., two red balls are in B_1 if and only if $x_1 = x_2 = 1$. Therefore, we can compute an AND value via $(\mathsf{Shuf}, \{1,2,4\})$ and $(\mathsf{TakeAll})$ in the same way to our two-input AND protocol.

Next, let us consider how to produce a commitment to $x_1 \wedge x_2$ from the above configurations (where B_3 is removed). Suppose that we repeat applying $(\mathsf{Shuf}, \{1,2,3\})$ and $(\mathsf{Take}, 1)$ until the color of a taken ball is red[v]. Then, the resulting configurations will be as follows:

$$\begin{aligned}
&(;\{\{\circ,\circ,\dots,\circ\}\},\{\{\bullet,\circ,\dots,\circ\}\},\{\{\circ,\circ,\dots,\circ\}\};[\bullet]) && (\tfrac{p_{00}}{2},\tfrac{p_{01}}{2},\tfrac{p_{10}}{2},0),\\
&(;\{\{\circ,\circ,\dots,\circ\}\},\{\{\circ,\circ,\dots,\circ\}\},\{\{\bullet,\circ,\dots,\circ\}\};[\bullet]) && (\tfrac{p_{00}}{2},\tfrac{p_{01}}{2},\tfrac{p_{10}}{2},0),\\
&(;\{\{\bullet,\circ,\dots,\circ\}\},\{\{\circ,\circ,\dots,\circ\}\},\{\{\circ,\circ,\dots,\circ\}\};[\bullet]) && (0,0,0,p_{11}),
\end{aligned} \tag{5.11}$$

where we omit the empty bag and empty trays. From the above configurations, notice that if we merge B_2 and B_3 into one bag, say B_2, then a pair of B_1 and B_2 can be a commitment to $x_1 \wedge x_2$. Thus, we can obtain the output commitment by moving all the balls in B_3 into B_2 (and put $2m$ white balls into B_1 to make the numbers of balls be the same).

Description. Given two commitments to $x_1 \in \{0,1\}$ and $x_2 \in \{0,1\}$, our committed-format AND protocol proceeds as shown in Protocol 5. We denote by B_1 and B_2 two bags constituting a commitment to x_1 and by B_3 and B_4 those constituting a commitment to x_2. Hereinafter, for successive two actions $(\mathsf{PublicPut}, \mathsf{b}_1, p)$ and $(\mathsf{PublicPut}, \mathsf{b}_2, p)$, we simply write $(\mathsf{PublicPut}, \mathsf{b}_1\mathsf{b}_2, p)$.

We omit a diagram of our committed-format AND protocol as it is similar to that of our multi-input AND protocol shown in Fig. 5.3. We discuss the performance of the protocol in Section 5.5.

[v] If it is $\circ$, we return $\circ$ into B_1.

5.4.2. How to Construct a Protocol for Any Function

See Eq. (5.10) again. We note that a committed-format NOT protocol can be easily constructed; just swapping two bags constituting a commitment to $x \in \{0, 1\}$ results in a commitment to the negation $\bar{x}$.

We now have committed-format AND and NOT protocols as shown above. Based on these protocols, we can construct a protocol for any Boolean function f as follows.

1. Create a Boolean circuit representing f (with AND and NOT gates).
2. Each player prepares a required number of commitments to his/her private input via PrivateInput according to the circuit.
3. Obtain a commitment to the output value of f by evaluating the circuit using our committed-format AND/NOT protocols. If the number of balls is different between input commitments when performing the AND protocol, players put white balls such that the number of balls becomes the same.

Since our committed-format AND protocol is secure and any information about the inputs and output does not leak (because of committed-format), the above protocol for f is also secure.

5.5. Performance of Our Protocols

This section discusses the efficiency of our AND protocols shown in Sections 5.2, 5.3, and 5.4.1. Table 5.1 summarizes the performance of our proposed protocols. We evaluated them in terms of two items: the number of balls and bags, and runtime.

The Number of Balls and Bags. Our two-input AND protocol requires six balls (namely, two red and four white balls) and three bags. Our conjecture is that two bags would be insufficient for a secure computation of the two-input AND computation. On the other hand, our AND protocol with n inputs requires $2n + 2$ balls (namely, two red and $2n$ white balls) and $n + 1$ bags. Our committed-format AND protocol would be inefficient because it requires $8m$ balls (and four bags) where m denotes the number of balls in each of bags constituting input commitments.

Let $\mathcal{C}$ be a Boolean circuit with the logical AND/NOT gates and $d(\mathcal{C})$ denote the depth of $\mathcal{C}$. Let us consider the number of required balls for our committed-format AND protocol that evaluates the "final" AND gate for $\mathcal{C}$ (i.e., it is of the depth $d(\mathcal{C})$), denoted by $\alpha_{d(\mathcal{C})}$. Remember that the number of balls in each of bags constituting the output commitment is four times greater than that of input commitments in our committed-format AND protocol. Thus, we have $\alpha_{d(\mathcal{C})} = 2 \times 4^{d(\mathcal{C})}$, i.e., $\alpha_{d(\mathcal{C})} = \mathcal{O}(4^{d(\mathcal{C})})$.

Table 5.1: The efficiency of our proposed protocols, where m denotes the number of balls in each of bags constituting an input commitment.

Function		#Balls	#Bags	Runtime
two-input AND	§5.2	6	3	Deterministic
n-input AND	§5.3	$2n+2$	$n+1$	Expected
committed-format AND	§5.4.1	$8m$	4	Expected
Boolean circuit $\mathcal{C}$	§5.4.2	$2 \times 4^{d(\mathcal{C})}$	$2 \times$ #input-node	Expected

Runtime. Our two-input AND protocol has a deterministic runtime. By contrast, our AND protocol with n inputs and committed-format AND protocol are Las Vegas algorithms. Each protocol includes a repetition of shuffling bags and taking a ball until • appears. Let us estimate the expected number of the repetition

For our AND protocol with n inputs, let us first consider the case of $n = 3$ as a simple case. Remember that we repeat $(\mathsf{Shuf}, \{1,2,3\})$ and $(\mathsf{Take}, 1)$ (and $(\mathsf{PublicPut}, \bullet, 1)$) from Eq. (5.4) until • appears. The expected number of the repetition is three because the probability of taking • is exactly 1/3. Then, let us consider the case of $n = 4$. In addition to the actions for $n = 3$, we repeat $(\mathsf{Shuf}, \{1,2,3,4\})$ and $(\mathsf{Take}, 1)$ from four bags including eight balls (namely, two red and six white balls) until • is taken. Therefore, the expected number of the repetition is $3 + 4 = 7$.

When we have n (≥ 3) inputs, the expected number of repetition can be written as follows:

$$\sum_{k=3}^{n} k = \frac{(n-2)(n+3)}{2}.$$

That is, it is $\mathcal{O}(n^2)$.

For our committed-format AND protocol, let us also show the expected number of repetition. The probability of taking • (as in Eq. (5.11)) is $2/(6m-2)$ because the total numbers of red and white balls in the bags are 2 and $6m-2$, respectively. Thus, the expected number of repetition is $3m-1$. As for the number of balls, let us consider the expected number of repetition for our committed-format AND protocol that evaluates the final AND gate for $\mathcal{C}$, denoted by $\beta_{d(\mathcal{C})}$. We have $\beta_{d(\mathcal{C})} = 3 \times 4^{d(C)-1} - 1$, i.e., $\beta_{d(\mathcal{C})} = \mathcal{O}(4^{d(\mathcal{C})})$.

5.6. Implementation Examples

In this section, we show implementation examples for ball-based cryptography to show the feasibility of using (physical) balls and bags. For this, we

Figure 5.4: Two-colored balls and three bags we purchased. The bags are closed at the top with drawstrings.

(a) PublicPut

(b) PrivatePut

(c) Shuf

(d) Take

(e) MergeBags

Figure 5.5: Implementing all the actions defined in Definition 2

purchased two-colored balls for approximately $1 and three bags for approximately $3 as shown in Fig. 5.4. Using these balls and bags, we implemented all the actions defined in Definition 2 (except for Move and Result[vi]), as shown in Fig. 5.5. As a result, we confirmed that ball-based cryptography proposed in this study is based on realistic physical operations and is feasible for humans.

5.7. Conclusion

In this work, we showed that balls and bags provide us a simple way for achieving a secure computation of the logical AND. We formalized protocols

[vi] Move can be performed without using a bag, and Result requires no ball and bag.

with balls and bags and showed how to construct a diagram of a protocol, from which its correctness and security can be confirmed. Moreover, we presented general MPCs by constructing committed-format AND and NOT protocols.

6. Online App

Cryptographic protocols using physical tools such as card-based cryptography enable us to easily achieve cryptographic tasks without the need for mathematical knowledge of algebra. However, all players participating in such a protocol are required to be in the same place because they use physical tools. In this chapter, we focus on how to perform such a protocol remotely while keeping its property (i.e., it can be performed easily) consistent. Precisely, we propose an auction protocol with a *messaging app* such as Facebook Messenger, which extends the existing protocol with envelopes [13]. Our idea is to utilize the read receipts and group chat features loaded in a messaging app. We note that security requirements between our proposed protocol and the existing protocol [13] are similar; while our proposed protocol requires a messaging app used in the protocol to have no vulnerability, the existing protocol also requires physical tools to have no vulnerability.

6.1. Proposed Protocol

Our proposed auction protocol computes in a similar way of [13] that it reveals bits from the most significant bit of bids one by one. The idea behind our proposed protocol is to make group chats with an auctioner and bidders per values of possible prices of bids.

6.2. Abstract

Let A denote an auctioner, $S_1, S_2, \ldots, S_n$ denote bidders, and $p_m > p_{m-1} > \cdots > p_1$ denote possible prices of bids.

Setup. The auctioner A makes m groups with the bidders $S_1, S_2, \ldots, S_n$ on a messaging app. Let p_i, $1 \leq i \leq m$, be the name of each group.

Bid. Let $p_{j'}$ denote the value of the bid by S_j, $1 \leq j \leq m$. The bidder S_j sends a message of 1 to the group of $p_{j'}$ and sends 0s to the remaining groups.

Open. The auctioner A read the messages sent to the group of p_m. If there is a message of 1, the bidder sent it becomes the winner, and A tells the winner and the highest value p_m to the bidders. Otherwise, A reads the messages sent to the one of p_{m-1}, and so on. If there is a message deleted by a bidder, A aborts the protocol.

Protocol 6. Message-app-based auction protocol.

1. **for** $i = 1$ to m **do**
2. $(\mathbf{Make}, p_i, A, \{S_1, \ldots, S_n\})$
3. **for** $i = 1$ to n **do**
4. $\left(\mathbf{Send}, S_i^{i'}, p_{i'}, S_i, (1, \emptyset)\right)$ where $p_{i'}$ denotes the bid of S_i
5. **for** $j = 1$ to m **do**
6. **if** $j \neq i'$ **then**
7. $\left(\mathbf{Send}, S_i^j, p_j, S_i, (0, \emptyset)\right)$
8. **for** $i = 0$ to $i = m - 1$ **do**
9. $(\mathbf{Read}, p_{m-i}, A)$
10. Let W be the set of sellers S_ℓ that performed $\left(\mathbf{Send}, S_\ell^{m-i}, p_{m-i}, S_\ell, (1, \emptyset)\right)$ for some $\ell \in \{1, \ldots, n\}$
11. **if** $W \neq \emptyset$ **then**
12. **for** $j = 1$ to m **do**
13. $(\mathbf{Tell}, A, S_j, W$ and $p_{m-i})$
14. A outputs W and p_{m-i}
15. Let k be $m - i$
16. **break**
17. **for** $i = 1$ to $i = n$ **do**
18. **for** $j = k$ to $j = m$ **do**
19. $(\mathbf{Read}, p_j, S_i)$
20. **if** there exists a seller S_ℓ that performed $\left(\mathbf{Send}, S_\ell^j, p_j, S_\ell, (1, \emptyset)\right)$ for some $\ell \in \{1, \ldots, n\}$ such that $j \neq k$ **then**
21. S_i outputs $\perp$
22. **for** $j = 1$ to $j = k - 1$ **do**
23. **if** $(\mathbf{Verify}, S_i^j, S_i) > 0$ **then**
24. S_i outputs $\perp$
25. **for** $i = 1$ to $i = n$ **do**
26. **for** $j = 1$ to $j = k - 1$ **do**
27. $(\mathbf{Delete}, S_i^j, S_i)$
28. S_i outputs W and p_k

Verification. Let p_h denote the highest value of the bids. The bidder S_j reads the messages sent to the groups of p_k for $k \geq h$ to verify the winner and p_h. Besides, S_j verifies that the messages S_j sent to the groupes of p_ℓ for $\ell < h$ have not been read. If not, S_j aborts the protocol.

Delete. The bidder S_j deletes the messages which S_j sent to the groupes of p_ℓ for $\ell < h$.

6.2.1. Formal Description

Here, we show a fomal description of our proposed auction protocol in Protocol 6, where for any $\mathsf{text} \in \mathcal{M}$, let $(\mathbf{Tell}, P_i, P_j, \mathsf{text})$ mean that P_i sends text to P_j by any means.

Bibliography

[1] Abe, Y., Iwamoto, M., Ohta, K.: Efficient private PEZ protocols for symmetric functions. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. Lecture Notes in Computer Science, vol. 11891, pp. 372–392. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-36030-6_15 7, 75

[2] Abe, Y., Hayashi, Y., Mizuki, T., Sone, H.: Five-card AND protocol in committed format using only practical shuffles. In: Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop. pp. 3–8. APKC '18, Association for Computing Machinery, New York, NY, USA (2018), https://doi.org/10.1145/3197507.3197510 7, 31, 58

[3] Araki, T., Furukawa, J., Lindell, Y., Nof, A., Ohara, K.: High-throughput semi-honest secure three-party computation with an honest majority. In: 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 805–817. CCS '16, Association for Computing Machinery, New York, NY, USA (2016), https://doi.org/10.1145/2976749.2978331 7

[4] Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. Theoretical Computer Science **306**(1), 69–84 (2003), https://doi.org/10.1016/S0304-3975(03)00210-X 7, 16, 75

[5] Bultel, X., Dreier, J., Dumas, J.G., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: Demaine, E.D., Grandoni, F. (eds.) Fun with Algorithms. Leibniz International Proceedings in Informatics (LIPIcs), vol. 49, pp. 8:1–8:20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2016), https://doi.org/10.4230/LIPIcs.FUN.2016.8 8, 32, 33

[6] Bultel, X., Dreier, J., Dumas, J., Lafourcade, P., Miyahara, D., Mizuki, T., Nagao, A., Sasaki, T., Shinagawa, K., Sone, H.: Physical zero-knowledge proof for Makaro. In: Stabilization, Safety, and Security of Distributed Systems. Lecture Notes in Computer Science, vol. 11201, pp. 111–125 (2018), https://doi.org/10.1007/978-3-030-03232-6_8 33

[7] Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology **1**(1), 65–75 (1988), https://doi.org/10.1007/BF00206326 72

[8] Chien, Y.F., Hon, W.K.: Cryptographic and physical zero-knowledge proof: From Sudoku to Nonogram. In: Boldi, P., Gargano, L. (eds.) Fun with Algorithms. Lecture Notes in Computer Science, vol. 6099, pp. 102–112. Springer, Berlin, Heidelberg (2010), https://doi.org/10.1007/978-3-642-13122-6_12 8, 32, 33

[9] Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) Advances in Cryptology—CRYPTO' 93. Lecture Notes in Computer Science, vol. 773, pp. 319–330. Springer, Berlin, Heidelberg (1994), https://doi.org/10.1007/3-540-48329-2_27 7, 58, 61, 67, 74

[10] D'Arco, P., De Prisco, R.: Secure computation without computers. Theoretical Computer Science **651**, 11–36 (2016), https://doi.org/10.1016/j.tcs.2016.08.003 7, 74

[11] Den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology—EUROCRYPT '89. Lecture Notes in Computer Science, vol. 434, pp. 208–217. Springer, Berlin, Heidelberg (1990), https://doi.org/10.1007/3-540-46885-4_23 7, 14, 61, 74, 75

[12] Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theor. **22**(6), 644–654 (2006), https://doi.org/10.1109/TIT.1976.1055638 7

[13] Dreier, J., Jonker, H., Lafourcade, P.: Secure auctions without cryptography. In: Ferro, A., Luccio, F., Widmayer, P. (eds.) Fun with Algorithms. Lecture Notes in Computer Science, vol. 8496, pp. 158–170. Springer, Cham (2014), https://doi.org/10.1007/978-3-319-07890-8_14 9, 94

[14] Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: Du, D.Z., Duan, Z., Tian, C. (eds.) Computing and Combinatorics. Lecture Notes in Computer Science, vol. 11653, pp. 166–177. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-26176-4_14 33

[15] Fagin, R., Naor, M., Winkler, P.: Comparing information without leaking it. Commun. ACM **39**(5), 77–85 (1996), https://doi.acm.org/10.1145/229459.229469 7, 16, 74

[16] Francis, D., Aljunid, S.R., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Necessary and sufficient numbers of cards for securely computing two-bit output functions. In: Phan, R.C.W., Yung, M. (eds.) Paradigms in Cryptology–Mycrypt 2016. Malicious and Exploratory Cryptology. Lecture Notes in Computer Science, vol. 10311, pp. 193–211. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-61273-7_10 8

[17] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. J. ACM **38**(3), 691–729 (1991), https://doi.acm.org/10.1145/116825.116852 35

[18] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989), https://doi.org/10.1137/0218012 32, 35

[19] Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. Theory of Computing Systems **44**(2), 245–268 (2009), https://doi.org/10.1007/s00224-008-9119-9 7, 8, 32, 33, 35, 36, 74

[20] Hanaoka, G.: Towards user-friendly cryptography. In: Phan, R.C.W., Yung, M. (eds.) Paradigms in Cryptology–Mycrypt 2016. Malicious and Exploratory Cryptology. Lecture Notes in Computer Science, vol. 10311, pp. 481–484. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-61273-7_24 31

[21] Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. In: Shikata, J. (ed.) Information Theoretic Security. Lecture Notes in Computer Science, vol. 10681, pp. 135–152. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-72089-0_8 62

[22] Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E101.A**(9), 1512–1524 (2018), https://doi.org/10.1587/transfun.E101.A.1512 8, 33, 38

[23] Ibaraki, T., Manabe, Y.: A more efficient card-based protocol for generating a random permutation without fixed points. In: Mathematics and Computers in Sciences and in Industry (MCSI). pp. 252–257 (2016), https://doi.org/10.1109/MCSI.2016.054 33, 38, 62

[24] Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Calude, C.S., Dinneen, M.J. (eds.) Unconventional Computation and Natural Computation. Lecture Notes in Computer Science, vol. 9252, pp. 215–226. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-21819-9_16 37, 61, 62

[25] Jakobsson, M., Yung, M.: Proving without knowing: On oblivious, agnostic and blindfolded provers. In: Koblitz, N. (ed.) Advances in Cryptology—CRYPTO'96. Lecture Notes in Computer Sci-

ence, vol. 1109, pp. 186–200. Springer, Berlin, Heidelberg (1996), https://doi.org/10.1007/3-540-68697-5_15 16

[26] Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology—ASIACRYPT 2017. Lecture Notes in Computer Science, vol. 10626, pp. 126–155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5 8

[27] Koch, A.: The landscape of optimal card-based protocols. Cryptology ePrint Archive, Report 2018/951 (2018), https://eprint.iacr.org/2018/951 8

[28] Koch, A.: Cryptographic Protocols from Physical Assumptions. Ph.D. thesis, Karlsruhe Institute of Technology (KIT) (2019), https://doi.org/10.5445/IR/1000097756 8

[29] Koch, A., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology–ASIACRYPT 2019. Lecture Notes in Computer Science, vol. 11921, pp. 488–517. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-34578-5_18 8, 31

[30] Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: Farach-Colton, M., Prencipe, G., Uehara, R. (eds.) Fun with Algorithms. Leibniz International Proceedings in Informatics (LIPIcs), vol. 157, pp. 17:1–17:23. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020), https://doi.org/10.4230/LIPIcs.FUN.2021.17 14, 18, 45, 47

[31] Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology—ASIACRYPT 2015. Lecture Notes in Computer Science, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-662-48797-6_32 8, 18, 31, 60, 65, 82

[32] Kölker, J.: Selected Slither Link variants are NP-complete. Journal of Information Processing **20**(3), 709–712 (2012), https://doi.org/10.2197/ipsjjip.20.709 43

[33] Komano, Y., Mizuki, T.: Multi-party computation based on physical coins. In: Fagan, D., Martín-Vide, C., O'Neill, M., Vega-Rodríguez, M.A. (eds.) Theory and Practice of Natural Computing. Lecture Notes in Computer Science, vol. 11324, pp. 87–98. Springer, Cham (2018), https://doi.org/10.1007/978-3-030-04070-3_7 7, 74

[34] Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: A physical ZKP for Slitherlink: How to perform physical topology-preserving computation. In: Heng, S.H., Lopez, J. (eds.) Information Security Practice and Experience. Lecture Notes in Computer Science, vol. 11879, pp. 135–151. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-34339-2_8 10

[35] Lafourcade, P., Mizuki, T., Nagao, A., Shinagawa, K.: Light cryptography. In: Drevin, L., Theocharidou, M. (eds.) Information Security Education. Education in Proactive Information Security. Advances in Information and Communication Technology, vol. 557, pp. 89–101. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-23451-5_7 7

[36] Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. Cryptology ePrint Archive, Report 2015/1031 (2015), https://eprint.iacr.org/2015/1031 31

[37] Miyahara, D., ichi Hayashi, Y., Mizuki, T., Sone, H.: Practical card-based implementations of Yao's millionaire protocol. Theoretical Computer Science **803**, 207–221 (2020), https://doi.org/10.1016/j.tcs.2019.11.005 9

[38] Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theor. Comput. Sci. **622**(C), 34–44 (2016), https://doi.org/10.1016/j.tcs.2016.01.039 7, 61

[39] Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: Foresti, S., Persiano, G. (eds.) Cryptology and Network Security. Lecture Notes in Computer Science, vol. 10052, pp. 484–499. Springer, Cham (2016), https://doi.org/10.1007/978-3-319-48965-0_29 7, 30, 31

[40] Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A.E. (eds.) Unconventional Computation and Natural Computation. Lecture Notes in Computer Science, vol. 7956, pp. 162–173. Springer, Berlin, Heidelberg (2013), https://doi.org/10.1007/978-3-642-39074-6_16 61

[41] Mizuki, T., Komano, Y.: Analysis of information leakage due to operative errors in card-based protocols. In: Iliopoulos, C., Leong, H.W., Sung, W.K. (eds.) Combinatorial Algorithms. Lecture Notes in Computer Science, vol. 10979, pp. 250–262. Springer, Cham (2018), https://doi.org/10.1007/978-3-319-94667-2_21 74, 82

[42] Mizuki, T., Kugimoto, Y., Sone, H.: Secure multiparty computations using a dial lock. In: Cai, J.Y., Cooper, S.B., Zhu, H. (eds.) Theory and

Applications of Models of Computation. Lecture Notes in Computer Science, vol. 4484, pp. 499–510. Springer, Berlin, Heidelberg (2007), https://doi.org/10.1007/978-3-540-72504-6_45 7

[43] Mizuki, T., Kugimoto, Y., Sone, H.: Secure multiparty computations using the 15 puzzle. In: Dress, A., Xu, Y., Zhu, B. (eds.) Combinatorial Optimization and Applications. Lecture Notes in Computer Science, vol. 4616, pp. 255–266. Springer, Berlin, Heidelberg (2007), https://doi.org/10.1007/978-3-540-73556-4_28 7, 16

[44] Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Wang, X., Sako, K. (eds.) Advances in Cryptology—ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 598–606. Springer, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-34961-4_36 7, 8, 61, 62, 74

[45] Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security **13**(1), 15–23 (2014), https://doi.org/10.1007/s10207-013-0219-4 18, 26, 60, 65, 75

[46] Mizuki, T., Shizuya, H.: Practical card-based cryptography. In: Ferro, A., Luccio, F., Widmayer, P. (eds.) Fun with Algorithms. Lecture Notes in Computer Science, vol. 8496, pp. 313–324. Springer, Cham (2014), https://doi.org/10.1007/978-3-319-07890-8_27 18, 33, 38, 61

[47] Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E100.A**(1), 3–11 (2017), https://doi.org/10.1587/transfun.E100.A.3 18

[48] Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics. Lecture Notes in Computer Science, vol. 5598, pp. 358–369. Springer, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-642-02270-8_36 7, 8, 14, 26, 27, 28, 29, 31, 58, 59, 60, 61, 62, 67, 68

[49] Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. The Australasian Journal of Combinatorics **36**, 279–293 (2006), https://ajc.maths.uq.edu.au/?page=get_volumes&volume=36 61

[50] Moran, T., Naor, M.: Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In: Vaudenay, S. (ed.) Advances in Cryptology—EUROCRYPT 2006. Lecture Notes in Computer

Science, vol. 4004, pp. 88–108. Springer, Berlin, Heidelberg (2006), https://doi.org/10.1007/11761679_7 7, 9, 74

[51] Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. Theoretical Computer Science **411**(10), 1283 – 1310 (2010), https://doi.org/10.1016/j.tcs.2009.10.023 7, 9, 74

[52] Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: How to solve millionaires' problem with two kinds of cards. New Generation Computing (2021, in press), https://doi.org/10.1007/s00354-020-00118-8 8, 13, 14, 15, 16, 26, 27, 28, 30

[53] Nakai, T., Shirouchi, S., Iwamoto, M., Ohta, K.: Four cards are sufficient for a card-based three-input voting protocol utilizing private permutations. In: Shikata, J. (ed.) Information Theoretic Security. Lecture Notes in Computer Science, vol. 10681, pp. 153–165. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-72089-0_9 16, 61

[54] Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations. In: Foresti, S., Persiano, G. (eds.) Cryptology and Network Security. Lecture Notes in Computer Science, vol. 10052, pp. 500–517. Springer, Cham (2016), https://doi.org/10.1007/978-3-319-48965-0_30 61

[55] Naor, M., Shamir, A.: Visual cryptography. In: De Santis, A. (ed.) Advances in Cryptology—EUROCRYPT'94. Lecture Notes in Computer Science, vol. 950, pp. 1–12. Springer, Berlin, Heidelberg (1995), https://doi.org/10.1007/BFb0053419 74

[56] Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theoretical Computer Science **191**(1–2), 173–183 (1998), https://doi.org/10.1016/S0304-3975(97)00107-2 7, 58

[57] Niemi, V., Renvall, A.: Solitaire zero-knowledge. Fundam. Inf. **38**(1,2), 181–188 (1999), https://doi.org/10.3233/FI-1999-381214 7, 31

[58] Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any Boolean function. In: Jain, R., Jain, S., Stephan, F. (eds.) Theory and Applications of Models of Computation. Lecture Notes in Computer Science, vol. 9076, pp. 110–121. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-17142-5_11 7, 61

[59] Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Securely computing three-input functions with eight cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E98.A**(6), 1145–1152 (2015), https://doi.org/10.1587/transfun.E98.A.1145 7, 61

[60] Nishida, T., Mizuki, T., Sone, H.: Securely computing the three-input majority function with eight cards. In: Dediu, A.H., Martín-Vide, C., Truthe, B., Vega-Rodríguez, M.A. (eds.) Theory and Practice of Natural Computing. Lecture Notes in Computer Science, vol. 8273, pp. 193–204. Springer, Berlin, Heidelberg (2013), https://doi.org/10.1007/978-3-642-45008-2_16 29

[61] Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101**(9), 1494–1502 (2018), https://doi.org/10.1587/transfun.E101.A.1494 23, 45

[62] Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Five-card secure computations using unequal division shuffle. In: Dediu, A.H., Magdalena, L., Martín-Vide, C. (eds.) Theory and Practice of Natural Computing. Lecture Notes in Computer Science, vol. 9477, pp. 109–120. Springer, Cham (2015), https://doi.org/10.1007/978-3-319-26841-5_9 31

[63] Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**, 361–371 (2018), https://doi.org/10.1007/s00500-017-2858-2 8, 62

[64] Ono, H., Manabe, Y.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: 2018 13th Asia Joint Conference on Information Security (AsiaJCIS). pp. 23–28 (2018), https://doi.org/10.1109/AsiaJCIS.2018.00013 16

[65] Ono, H., Manabe, Y.: Card-based cryptographic protocols with the minimum number of rounds using private operations. In: Pérez-Solà, C., Navarro-Arribas, G., Biryukov, A., Garcia-Alfaro, J. (eds.) Data Privacy Management, Cryptocurrencies and Blockchain Technology. Lecture Notes in Computer Science, vol. 11737, pp. 156–173. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-31500-9_10 16

[66] Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Generation Computing (2020, in press), https://doi.org/10.1007/s00354-020-00113-z 16

[67] Ruangwises, S., Itoh, T.: AND protocols using only uniform shuffles. In: van Bevern, R., Kucherov, G. (eds.) Computer Science–Theory and Applications. Lecture Notes in Computer Science, vol. 11532, pp. 349–358. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-19955-5_30 7

[68] Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for connected spanning subgraph problem and Bridges puzzle (2020), https://arxiv.org/abs/2011.02313 9

[69] Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Numberlink puzzle and k vertex-disjoint paths problem. New Generation Computing (2020, in press), https://doi.org/10.1007/s00354-020-00114-y 9

[70] Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Ripple Effect. In: Hong, S., Nandy, S., Uehara, R. (eds.) WALCOM: Algorithms and Computation. Lecture Notes in Computer Science, vol. 11737. Springer, Cham (2021, in press) 9

[71] Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for Sudoku. Theoretical Computer Science **839**, 135–142 (2020), https://doi.org/10.1016/j.tcs.2020.05.036 10

[72] Sasaki, T., Mizuki, T., Sone, H.: Card-based zero-knowledge proof for Sudoku. In: Ito, H., Leonardi, S., Pagli, L., Prencipe, G. (eds.) Fun with Algorithms. Leibniz International Proceedings in Informatics (LIPIcs), vol. 100, pp. 29:1–29:10. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018), https://doi.org/10.4230/LIPIcs.FUN.2018.29 33, 45

[73] Shinagawa, K.: On the Construction of Easy to Perform Card-Based Protocols. Ph.D. thesis, Tokyo Institute of Technology (2020) 8

[74] Shinagawa, K.: Card-based cryptography with dihedral symmetry. New Generation Computing (2021, in press), https://doi.org/10.1007/s00354-020-00117-9 8

[75] Shinagawa, K., Mizuki, T., Schuldt, J., Nuida, K., Kanayama, N., Nishide, T., Hanaoka, G., Okamoto, E.: Secure computation protocols using polarizing cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E99.A**(6), 1122–1131 (2016), https://doi.org/10.1587/transfun.E99.A.1122 7

[76] Shinagawa, K., Mizuki, T., Schuldt, J., Nuida, K., Kanayama, N., Nishide, T., Hanaoka, G., Okamoto, E.: Card-based protocols using regular polygon cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E100.A**(9), 1900–1909 (2017), https://doi.org/10.1587/transfun.E100.A.1900 8

[77] Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any Boolean circuit. Discrete Applied Mathematics **289**, 248–261 (2021), https://doi.org/10.1016/j.dam.2020.10.013 7

[78] Stiglic, A.: Computations with a deck of cards. Theoretical Computer Science **259**(1–2), 671–678 (2001), https://doi.org/10.1016/S0304-3975(00)00409-6 7, 58, 67

[79] Toyoda, K., Miyahara, D., Mizuki, T., Sone, H.: Six-card finite-runtime XOR protocol with only random cut. In: Proceedings of the 7th ACM Workshop on ASIA Public-Key Cryptography. pp. 2–8. APKC '20, Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3384940.3388961 7

[80] Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. International Journal of Information Security **19**(4), 445–452 (2020), https://doi.org/10.1007/s10207-019-00463-w 10

[81] Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Martín-Vide, C., Mizuki, T., Vega-Rodríguez, M.A. (eds.) Theory and Practice of Natural Computing. Lecture Notes in Computer Science, vol. 10071, pp. 58–69. Springer, Cham (2016), https://doi.org/10.1007/978-3-319-49001-4_5 14, 17, 31, 70

[82] Watanabe, Y., Kuroki, Y., Suzuki, S., Koga, Y., Iwamoto, M., Ohta, K.: Card-based majority voting protocols with three inputs using three cards. In: 2018 International Symposium on Information Theory and Its Applications (ISITA). pp. 218–222 (2018), https://doi.org/10.23919/ISITA.2018.8664324 16

[83] Yao, A.C.: Protocols for secure computations. In: Foundations of Computer Science. pp. 160–164. IEEE Computer Society, Washington, DC, USA (1982), https://doi.org/10.1109/SFCS.1982.88 7, 8, 13, 74

[84] Yato, T., Seta, T.: Complexity and completeness of finding another solution and its application to puzzles. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **86-A**(5), 1052–1060 (2003), https://search.ieice.org/bin/summary.php?id=e86-a_5_1052 35, 43

List of Publications

Journal Papers

1. Ken Takashima, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. "Actively Revealing Card Attack on Card-based Protocols." *Natural Computing*, Springer, accepted. DOI: 10.1007/s11047-020-09838-8.

2. Daiki Miyahara, Itaru Ueda, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Evaluating Card-based Protocols in Terms of Execution." *International Journal of Information Security*, Springer, in press. DOI: 10.1007/s10207-020-00525-4.

3. Ken Takashima, Yuta Abe, Tatsuya Sasaki, Daiki Miyahara, Kazumasa Shinagawa, Takaaki Mizuki, and Hideaki Sone. "Card-based Protocols for Secure Ranking Computations." *Theoretical Computer Science*, Elsevier, vol.845, pp.122–135, 2020. DOI: j.tcs.2020.09.008.

4. Tatsuya Sasaki, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. "Efficient Card-based Zero-Knowledge Proof for Sudoku." *Theoretical Computer Science*, Elsevier, vol.839, pp.135–142, 2020. DOI: j.tcs.2020.05.036.

5. Itaru Ueda, Daiki Miyahara, Akihiro Nishimura, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Secure Implementations of a Random Bisection Cut." *International Journal of Information Security*, Springer, vol.19, pp.445–452, 2020. DOI: 10.1007/s10207-019-00463-w.

6. Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Practical Card-based Implementations of Yao's Millionaire Protocol." *Theoretical Computer Science*, Elsevier, vol.803, pp.207–221, 2020. DOI: j.tcs.2019.11.005.

7. Daiki Miyahara, Tatsuya Sasaki, Takaaki Mizuki, and Hideaki Sone. "Card-based Physical Zero-knowledge Proof for Kakuro." *IEICE Trans. Fundamentals*, vol.E102-A, no.9, pp.1072–1078, 2019. DOI: 10.1587/transfun.E102.A.1072.

International Conferences

8. Soma Murata, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. "Efficient Generation of a Card-based Uniformly Distributed Random

Derangement". *15th International Conference and Workshop on Algorithms and Computation* (WALCOM 2021), Lecture Notes in Computer Science, Springer, vol.12635, in press.

9. Yuto Shinoda, Daiki Miyahara, Kazumasa Shinagawa, Takaaki Mizuki, and Hideaki Sone. "Card-based Covert Lottery". *13th International Conference on Security for Information Technology and Communications* (SECITC 2020), Lecture Notes in Computer Science, Springer, in press.

10. Léo Robert, Daiki Miyahara, Pascal Lafourcade, and Takaaki Mizuki. "Physical Zero-Knowledge Proof for Suguru Puzzle". *22nd International Symposium on Stabilization, Safety, and Security of Distributed Systems* (SSS 2020), Lecture Notes in Computer Science, Springer, vol.12514, pp.235–247, 2020. DOI: 978-3-030-64348-5_19.

11. Takahiro Saito, Daiki Miyahara, Takaaki Mizuki, and Hiroki Shizuya. "How to Implement a Non-uniform or Non-closed Shuffle". *9th International Conference on the Theory and Practice of Natural Computing* (TPNC 2020), Lecture Notes in Computer Science, Springer, vol.12494, pp.107–118, 2020. DOI: 10.1007/978-3-030-63000-3_9.

12. Soma Murata, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. "Public-PEZ Cryptography". *23rd Information Security Conference* (ISC 2020), Lecture Notes in Computer Science, Springer, vol.12472, pp.59–74, 2020. DOI: 10.1007/978-3-030-62974-8_4.

13. Daiki Miyahara, Léo Robert, Pascal Lafourcade, So Takeshige, Takaaki Mizuki, Kazumasa Shinagawa, Atsuki Nagao, and Hideaki Sone. "Card-based ZKP Protocols for Takuzu and Juosan". *10th International Conference on FUN with Algorithms* (FUN 2021), Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol.157, pp.20:1–20:21, 2020. DOI: 10.4230/LIPIcs.FUN.2021.20.

14. Kodai Toyoda, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. "Six-Card Finite-Runtime XOR Protocol with Only Random Cut". *7th ACM International Workshop on ASIA Public-Key Cryptography* (APKC 2020), ACM, pp.2–8, 2020. DOI: 10.1145/3384940.3388961.

15. Ken Takashima, Yuta Abe, Tatsuya Sasaki, Daiki Miyahara, Kazumasa Shinagawa, Takaaki Mizuki, and Hideaki Sone. "Card-based Secure Ranking Computations". *13th Annual International Conference on Combinatorial Optimization and Applications* (COCOA 2019), Lecture Notes in Computer Science, Springer, vol.11949, pp.461–472, 2019. DOI: 10.1007/978-3-030-36412-0_37.

16. Ken Takashima, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. "Card-based Protocol against Actively Revealing Cards Attack". *8th International Conference on the Theory and Practice of Natural Computing* (TPNC 2019), Lecture Notes in Computer Science, Springer, vol.11934, pp.95–106, 2019. DOI: 10.1007/978-3-030-34500-6_6.

17. Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Tatsuya Sasaki, and Hideaki Sone. "A Physical ZKP for Slitherlink: How to Perform Physical Topology-Preserving Computation". *15th International Conference on Information Security Practice and Experience* (ISPEC 2019), Lecture Notes in Computer Science, Springer, vol.11879, pp.135–151, 2019. DOI: 10.1007/978-3-030-34339-2_8.

18. Jean-Guillaume Dumas, Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Tatsuya Sasaki, and Hideaki Sone. "Interactive Physical Zero-Knowledge Proof for Norinori *25th International Computing and Combinatorics Conference* (COCOON 2019), Lecture Notes in Computer Science, Springer, vol.11653, pp.166–177, 2019. DOI: 10.1007/978-3-030-26176-4_14.

19. Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Practical and Easy-to-Understand Card-based Implementation of Yao's Millionaire Protocol". *12th Annual International Conference on Combinatorial Optimization and Applications* (COCOA 2018), Lecture Notes in Computer Science, Springer, vol.11346, pp.246–261, 2018. DOI: 10.1007/978-3-030-04651-4_17.

20. 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". *20th International Symposium on Stabilization, Safety, and Security of Distributed Systems* (SSS 2018), Lecture Notes in Computer Science, Springer, vol.11201, pp.111–125, 2018. DOI: 10.1007/978-3-030-03232-6_8.

21. Daiki Miyahara, Itaru Ueda, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. "Analyzing Execution Time of Card-based Protocols". *17th International Conference on Unconventional Computation and Natural Computation* (UCNC 2018), Lecture Notes in Computer Science, Springer, vol.10867, pp.145–158, 2018. DOI: 10.1007/978-3-319-92435-9_11.

22. 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". *23rd Annual*

International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2017), Lecture Notes in Computer Science, Springer, vol.10626, pp.126–155, 2017. DOI: 10.1007/978-3-319-70700-6_5.