PAPER *Special Section on Discrete Mathematics and Its Applications*

Secure Computation Protocols Using Polarizing Cards*

Kazumasa SHINAGAWA[†,††a)], ***Nonmember***, **Takaaki MIZUKI**[†††], ***Member***, **Jacob C. N. SCHULDT**[††], **Koji NUIDA**[††], ***Nonmembers***, **Naoki KANAYAMA**[†], **Takashi NISHIDE**[†], **Goichiro HANAOKA**[††], ***Members***, ***and*** **Eiji OKAMOTO**[†], ***Fellow***

SUMMARY It is known that, using just a deck of cards, an arbitrary number of parties with private inputs can securely compute the output of any function of their inputs. In 2009, Mizuki and Sone constructed a six-card COPY protocol, a four-card XOR protocol, and a six-card AND protocol, based on a commonly used encoding scheme in which each input bit is encoded using two cards. However, up until now, there are no known results to construct a set of COPY, XOR, and AND protocols based on a two-card-per-bit encoding scheme, which all can be implemented using only four cards. In this paper, we show that it is possible to construct four-card COPY, XOR, and AND protocols using polarizing plates as cards and a corresponding two-card-per-bit encoding scheme. Our protocols use a minimum number of cards in the setting of two-card-per-bit encoding schemes since four cards are always required to encode the inputs. Moreover, we show that it is possible to construct two-card COPY, two-card XOR, and three-card AND protocols based on a one-card-per-bit encoding scheme using a common reference polarizer which is a polarizing material accessible to all parties.

key words: *card-based protocols, polarizing cards, secure computation, boolean circuits, recreational cryptography*

1. Introduction

1.1 Background

Secure Multi-Party Computation (MPC) enables an arbitrary number of parties with secret inputs to compute the output of a function without revealing their inputs. MPC protocols are usually implemented in a computer-based environment. On the other hand, there are various works constructing MPC protocols using only a deck of physical cards, referred to as *card-based protocols* [1]–[17]. Compared to computer-based protocols, a card-based protocol has several advantages; (1) It is easy to understand the correctness and security of protocols even for non-experts, (2) Since card-based protocols do not rely on computers, they can be performed without the use of electricity, (3) In contrast to online protocols where players are invisible to each other, there is a potentially high cost of acting maliciously in card-based protocols, (4) Playing a card-based protocol is a lot of fun!

The *Five-Card Trick* proposed by den Boer [1] in 1989 is the first card-based protocol, which enables two parties to securely compute the AND ($\wedge$) function of their secret inputs using five cards (two $\boxed{\clubsuit}$s and three $\boxed{\heartsuit}$s). In [1] (as well as in previous works [1]–[14], [17]), a binary input is encoded by two cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0, \boxed{\heartsuit}\boxed{\clubsuit} = 1$. We refer to such an encoding using two cards as a *two-card-per-bit encoding* scheme. In the input phase, a party encodes his input as above, and places the cards face down in a public space (e.g. on a table). This pair of cards is called a *commitment*. As subsequent works, there are many works to reduce the number of cards in card-based protocols whose outputs are commitments. (The Five-Card Trick [1] does not enjoy this property since it outputs the value of $x \wedge y$.) The state-of-the-art protocols are summarized as follows: In the above encoding scheme ($\boxed{\clubsuit}\boxed{\heartsuit} = 0, \boxed{\heartsuit}\boxed{\clubsuit} = 1$), Mizuki and Sone [9] constructed a six-card COPY protocol and a four-card XOR protocol, and Koch [9], and Koch, Walzer and Härtel [4] constructed a five-card AND protocol; see Table 1. There are a five-card COPY protocol [13] and a four-card AND protocol [4], however, these protocols are Las-Vegas protocols, i.e., their computational times are finite only in expectation. On the other hand, in this paper, we focus on protocols which always terminate in finite runtime.

In this paper, we focus on the number of cards, in particular, *minimality*. We say that a card-based protocol is *minimal* with respect to an encoding scheme if it is impossible to construct a protocol providing the same functionality

Table 1 Previous card-based protocols (with finite runtime).

	Type of cards	# of cards	Minimality
∘ COPY protocol			
Mizuki-Sone [9]	♣, ♡	6	?
∘ XOR protocol			
Mizuki-Sone [9]	♣, ♡	4	✓
∘ AND protocol			
Mizuki-Sone [9]	♣, ♡	6	×
Koch-Walzer-Härtel [4]	♣, ♡	5	✓
∘ Input-preserving XOR protocol			
Nishida et al. [12]	♣, ♡	6	?
∘ Input-preserving AND protocol			
Nishida et al. [12]	♣, ♡	6	?
∘ Protocol for an arbitrary n-variable function			
Nishida et al. [12]	♣, ♡	$2n + 6$	?

Manuscript received September 18, 2015.
Manuscript revised February 22, 2016.

[†]The authors are with University of Tsukuba, Tuskuba-shi, 305-8577 Japan.
[††]The authors are with National Institute of Advanced Industrial Science and Technology, Tokyo, 135-0064 Japan.
[†††]The author is with Tohoku University, Sendai-shi, 980-8578 Japan.

[*]A preliminary conference version of this paper appeared at [15]. Three main additions to the conference version are listed as: covers in Sect. 2.1, detail constructions in Sect. 6.1, and protocol for an arbitrary function in Sect. 6.2.

a) E-mail: shinagawa@cipher.risk.tsukuba.ac.jp

DOI: 10.1587/transfun.E99.A.1122

using fewer cards; see Definition 8. We note that minimality depends on the underlying encoding scheme. From this point of view, the four-card XOR protocol [9] is minimal with respect to a two-card-per-bit encoding scheme since it takes four cards as input. On the other hand, there are no known minimal COPY/AND protocols, (cf. the state-of-the-art is the six-card COPY/AND protocols [9]). We emphasize that in particular a minimal COPY protocol is desired in the construction of more general protocols. Consider computing a given function $f(x_0, x_1, \cdots, x_{n-1})$ from commitments to $x_0, x_1, \cdots, x_{n-1} \in \{0, 1\}$. In many cases, it is required to preserve the input commitments since they might be needed in later stages of the protocol. To achieve this, the commitments are copied using the COPY protocol before evaluation of $f(x_0, x_1, \cdots, x_{n-1})$. If the commitments are copied in parallel, and the COPY protocol requires two additional cards besides the cards for storing the commitments, $2n$ additional cards are required in total[†]. In addition, reducing the number of cards leads to better performance, and thereby improves the viability of card-based protocols in the real world. Thus, constructing optimal protocols are important not only from a theoretical viewpoint but also from a practical viewpoint.

Unfortunately, it seems impossible to construct a minimal COPY protocol using previous cards because, to ensure correctness, it is necessary to open a face-down card, but the number of face-down cards is less than four after this opening. Moreover, [4] showed that there is no (finite-runtime) four-card AND protocol, i.e., the additional cards are essentially needed in secure computation of AND function. Due to these impossibility results, it seems to be necessary to introduce new frameworks.

1.2 Our Contribution

In this paper, we show that it is possible to construct minimal four-card COPY/XOR/AND protocols, by using *polarizing plates* as cards and a corresponding two-card-per-bit encoding. The use of polarizing plates enables us to overcome the limitations of ordinary cards. As an additional technical contribution we introduce a new shuffle, a *diagonal flip shuffle*, which is easily implementable and enables the construction of our four-card AND protocol. As applications of our protocols, we show minimal constructions of input-preserving XOR and AND protocols, which we combine to obtain a minimal half-adder protocol, a minimal full-adder protocol, a minimal voting protocol, all using a two-card-per-bit encoding scheme.

(1) Polarizing Cards

Polarization of light is a physical phenomenon that transforms light waves oscillating in various directions into light waves oscillating in a certain direction. A polarizing plate is a material which has such a polarization property. In

[†]If the commitments are copied one-by-one, only two additional cards are needed, but this is inefficient with respect to required computational time.

Table 2 Summarize of our protocols.

	Type of cards	# of cards	Minimality
◦ COPY protocol			
Sect. 4.1	Polarizing	4	✓
◦ XOR protocol			
Sect. 4.2	Polarizing	4	✓
◦ AND protocol			
Sect. 4.3	Polarizing	4	✓
◦ Input-preserving XOR protocol			
Sect. 5 (1)	Polarizing	4	✓
◦ Input-preserving AND protocol			
Sect. 5 (2)	Polarizing	4	✓
◦ Protocol for an arbitrary n-variable function			
Sect. 6.2	Polarizing	$2n + 2$	✓

Sect. 2.1, we define a polarizing card by a square polarizing plate with either vertical ($\updownarrow$) or horizontal ($\leftrightarrow$) direction. To encode 0, we use a sequence of two cards with the same direction, i.e., ($\updownarrow, \updownarrow$) or ($\leftrightarrow, \leftrightarrow$). Likewise, to encode 1, we use a sequence of two cards with opposite directions, i.e., ($\updownarrow, \leftrightarrow$) or ($\leftrightarrow, \updownarrow$). This sequence is called a *commitment*. The commitments of 0 and 1 are indistinguishable without superimposing the cards of the commitment sequence.

(2) COPY/XOR/AND Protocols

In Sect. 4, under the above encoding scheme, we construct minimal COPY/XOR/AND protocols in which only four cards are required (see Table 2). Our protocols use a rotation shuffle and a diagonal flip shuffle (the latter is only used in our AND protocol), which are defined in Sect. 2.2. We also show that our protocols are secure in the sense of Definition 7, in Sect. 3.2.

(3) Input-Preserving Protocols

We say a protocol for $f(x, y)$ is *input-preserving* if it outputs commitments to $f(x, y)$ and x (or y). In Sect. 5, we construct minimal input-preserving XOR/AND protocols (see Table 2). As applications of these protocols, applying the techniques from [5], [12], we also show the construction of a minimal half-adder protocol, a minimal full-adder protocol, a minimal voting protocol.

(4) Protocols Using a One-Card-Per-Bit Encoding

Using a *common reference polarizer*, binary information can be encoded using a single card per bit (referred to as a *one-card-per-bit encoding scheme*). In Sect. 6, we construct elementary protocols based on this type of encoding scheme. Utilizing these protocols in combination with a two-card-per-bit encoding scheme, we futhermore construct a minimal inout-preserving protocol for an arbitrary boolean function (note that the minimality of the protocol is with respect to a two-card-per-bit encoding scheme). We emphasize that the two-card-per-bit encoding and the one-card-per-bit encoding require different and incomparable setups. In particular, in the one-card-per-bit encoding scheme, parties cannot make commitments without access to the common reference polarizer. This is inconvenient if parties want to make commitments to their inputs prior to the execution of the protocol. As in previous works, our main aim is to construct protocols in the two-card-per-bit encoding scheme which enables parties to encode inputs locally at any time.

1.3 Related Works

In 1989, using two types of cards ($\boxed{\clubsuit}$, $\boxed{\heartsuit}$), den Boer [1] proposed a five-card AND protocol that reveals the output value at the end of the protocol. Most of subsequent works use the same type of cards as in [1]. In 2014, Mizuki-Shizuya [8] discussed the advantages and disadvantages of a one-card-per-bit encoding scheme that enables the construction of a three-card AND protocol, a two-card XOR protocol, and a three-card COPY protocol. A possible application of a similar encoding scheme to our polarizing cards is discussed in Sect. 6.

Based on elementary protocols, many applications are proposed [5], [11], [12]. In 2013, Mizuki, Asiedu, and Sone [5] constructed an eight-card half-adder protocol, a ten-card full-adder protocol, and a $(2\lceil\log n\rceil+6)$-card voting protocol where n is the number of parties. In 2015, Nishida et al. [12] constructed a six-card AND protocol with input-preserving property, which also outputs one of the input commitments. They also show that it is possible to construct a six-card half-adder protocol and protocols for any k-variable functions using $2k + 6$ cards and any k-variable symmetric functions using $2k + 2$ cards [12]. The numbers of cards for such advanced protocols are also reduced due to our improvements mentioned above.

In 2015, Nishimura et al. constructed a five-card Las-Vegas COPY protocol using unequal division shuffle [13]. In 2015, Koch, Walzer and Härtel [4] constructed a four-card Las-Vegas AND protocol, and a five-card AND protocol. They are very interesting from theoretical viewpoint, however, our focus is protocols which always terminate in finite runtime.

2. Polarizing Cards

In this section, we first propose a card called a polarizing card, which is a square polarizing plate. Next, we define three operations, permute, superimpose, and shuffle, that can be applied to a sequence of cards in a protocol. Furthermore, we propose two concrete shuffles, a *rotation shuffle* and a *diagonal flip shuffle*.

2.1 Polarizing Cards

(1) Polarizing Cards

Polarization of light is a physical phenomenon that transforms light waves oscillating in various directions into light waves oscillating in a certain direction. A polarizing plate is a material which has such a polarization property. If two polarizing plates with a same direction are superimposed, then light will pass through the plates which will appear to be transparent. On the other hand, if two plates with perpendicular directions are superimposed, no light will pass through

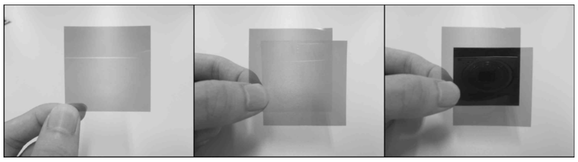


Fig. 1 Polarizing cards.

the plates which will appear to be black. In addition, a polarizing plate has an important property, which we will refer to as a *direction indistinguishability*. Specifically, it is difficult to distinguish the direction of the polarization without superimposing the plate with another polarizing plate.

We say that a polarizing plate is a *polarizing card* if its polarizing direction is either vertical or horizontal direction, and it has two symmetries, 90° *rotational symmetry* and *two-sided symmetry*, i.e., the face of the plate is invariant even if it is rotated 90° or flipped (e.g. square polarizing plate as shown on the left in Fig. 1). We denote by $\mathcal{D} = \{\updownarrow, \leftrightarrow\}$ the set of polarizing cards. If two cards with the same direction ($\updownarrow, \updownarrow$ or $\leftrightarrow, \leftrightarrow$) are superimposed, then light can pass through the plates and they appear transparent. We say that this result is "*white*" (Fig. 1, center). On the other hand, if two plates with perpendicular directions ($\updownarrow, \leftrightarrow$ or $\leftrightarrow, \updownarrow$) are superimposed, no light can pass through the plates. We say that this result is "*black*" (Fig. 1, right). A k-sequence of polarizing cards is cards lying in the public space such that each of the cards is isolated and not superimposed on top of each other or any other polarization plates. We denote by $(a_0, a_1, \cdots, a_{k-1})$ a sequence of polarizing cards.

(2) New Encoding

How should we encode $x \in \{0, 1\}$ using polarizing cards? One possibility is to use the previous encoding scheme from [1], i.e., $(\updownarrow, \leftrightarrow) = 0$ and $(\leftrightarrow, \updownarrow) = 1$. However, it is difficult to encode and decode due to the direction indistinguishability. For this reason, we propose a new encoding scheme, which is explicitly based on the polarizing property of the cards. The new encoding Enc is a function that maps $x \in \{0, 1\}$ to a sequence of two cards. Specifically, to encode 0, we use a sequence of two cards with the same direction, which will result in white when superimposed, i.e., $\mathsf{Enc}(0) = \{(\updownarrow, \updownarrow), (\leftrightarrow, \leftrightarrow)\}$. Likewise, to encode 1, we use a sequence of two cards with opposite directions, which will result in black when superimposed, i.e., $\mathsf{Enc}(1) = \{(\updownarrow, \leftrightarrow) = (\leftrightarrow, \updownarrow)\}$[†]. We say that a sequence is a *commitment* to a when it is an element of $\mathsf{Enc}(a)$. Note that, due to the direction indistinguishability property, nobody can distinguish whether a commitment belongs to $\mathsf{Enc}(0)$ or $\mathsf{Enc}(1)$ without superimposing the used cards. From now on, in order to simplify the exposition, we denote $\mathcal{D}$ by $\{0, 1\}$ instead of $\{\updownarrow, \leftrightarrow\}$. Using this notation, a commitment to x can be represented as $(c, c \oplus x)$, and this is equivalent to $(c' \oplus x, c')$ where $c = c' \oplus x$.

[†]The other possibility is to use the encoding scheme such that $\mathsf{Enc}(0) = \{(\updownarrow, \leftrightarrow), (\leftrightarrow, \updownarrow)\}$ and $\mathsf{Enc}(1) = \{(\updownarrow, \updownarrow), (\leftrightarrow, \leftrightarrow)\}$. They are equivalent.

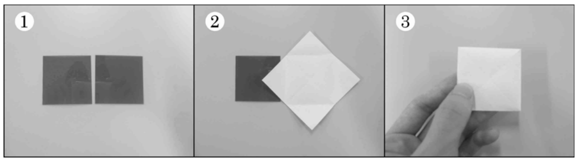


Fig. 2 Cover: paper cover with rotation symmetry.

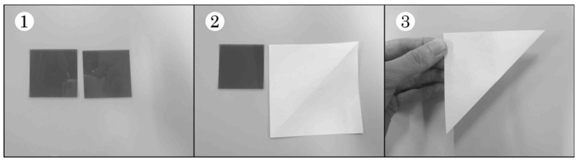


Fig. 3 Cover: paper cover with diagonal flip symmetry.

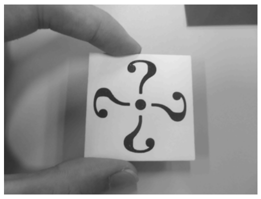


Fig. 4 Cover: non-transparent card.

(3) Covers

Covers are used in shuffles to avoid revealing partial information relevant to the secret value of the commitments. For example, when two cards $(x \oplus r, y \oplus r)$ are generated from (x, y) by applying the rotation shuffle in Sect. 2.2, two cards are stacked and rotated a random number of times. However, if two cards are just stacked, we learn whether the superimposing result is black or white, i.e., whether $x = y$ or not. To perform rotation/diagonal-flip shuffles securely, covers have rotational symmetry or diagonal flip symmetry. This requirement can be easily satisfied by using a square piece of paper as in Figs. 2 and 3. As an alternative solution, a non-transparent card in Fig. 4 can be used as a cover. However, the diagonal flip shuffle needs two non-transparent cards to cover the front and back sides.

2.2 Operations

(1) **Permutation**

In a previous work [7], the group of permutations for sequences of k cards is set to be the symmetric group S_k, which acts on each sequence by changing the order of cards. In this paper, we extend the definition of permutations for cards to deal with rotations of each card in a sequence (which is typical for our polarizing cards) as well as the re-ordering. We define permutation operations as follows.

Definition 1: Let Σ_k be the group of (extended) permutations generated by σ_i for $i = 0, 1, \ldots, k-2$ and ρ_i for $i = 0, 1, \ldots, k-1$, where σ_i and ρ_i denote the *adjacent transposition* and the *fundamental rotation*, respectively, defined as follows:

$$\sigma_i(x_0, \cdots, x_i, x_{i+1}, \cdots, x_{k-1}) = (x_0, \cdots, x_{i+1}, x_i, \cdots, x_{k-1})$$
$$\rho_i(x_0, \cdots, x_i, \cdots, x_{k-1}) = (x_0, \cdots, \overline{x_i}, \cdots, x_{k-1}).$$

A *permutation operation*, defined by $\sigma \in \Sigma_k$, takes as input a sequence $s = (x_0, \ldots, x_{k-1})$, and outputs $\sigma(s)$.

(2) **Superimposing**

We define superimposing operations as follows.

Definition 2: We define a *superimposing operation* that takes as inputs $s = (x_0, \cdots, x_{k-1})$ and i, j where $i, j \in \{0, \cdots, k-1\}$ are different indices ($i \neq j$), and outputs "*white*" if $x_i = x_j$ or "*black*" if $x_i \neq x_j$.

This is physically implemented by superimposing cards x_i and x_j. This operation is similar to an *opening operation* that is used in ordinary card-based protocols. However, a superimposing operation to x, y reveals only the direction difference between x, y, i.e., $x \oplus y$, while an opening operation reveals the value of card itself.

(3) **Shuffle**

We define shuffle operations that play an important role in achieving perfect secrecy in card-based protocols. Firstly, we provide a general definition of shuffles that might contain a shuffle which has no easy physical implementation:

Definition 3: Let $\sigma \in \Sigma_k$ be a permutation with order ℓ, i.e., $\sigma^\ell = id$ where id is the identity mapping. We define a *shuffle operation* based on σ that takes $s = (x_0, \cdots, x_{k-1})$ as input, and outputs $\sigma^r(s)$, where r is uniformly chosen from $\{0, \cdots, \ell - 1\}$. The value r is assumed to be hidden from the parties executing the protocol.

We show two implementations of a shuffle based on σ. The first one is simple. A party applies the permutation σ to s a random number of times until all of parties are satisfied. The random number is not known to either of them even the party who operates the shuffle. The other method does not require a party who takes responsibility of security. Let $P_0, \cdots, P_{n-1}$ be the parties participating in the protocol. P_0 uniformly chooses $r_0 \in \{0, \cdots, \ell - 1\}$ where ℓ is the order of σ, operates permutation σ^{r_0} to s, and sends $\sigma^{r_0}(s)$ to P_1. This is repeated from P_0 to P_{n-1}. Finally, P_{n-1} places $\sigma^r(s)$ in a public space where $r = r_0 + \cdots + r_{n-1}$. Obviously, the value r is uniformly chosen from $\{0, \cdots, \ell - 1\}$ and nobody knows it. In practice, the former method is usually used since it is easy to demonstrate. Therefore, it is important that a shuffle is easily implemented by hand. In this paper, we say that a shuffle has an *easy physical implementation* if it is achieved only using "one operation" that can be performed by hand.

We propose two shuffles, a *rotation shuffle* and a *diagonal flip shuffle*. These two shuffles have an easy physical implementation. The idea of a rotation shuffle is used in the protocols [8], but we define it more formally:

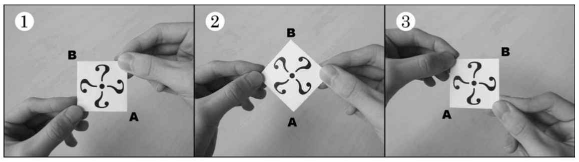


Fig. 5 An implementation of the rotation operation ρ.

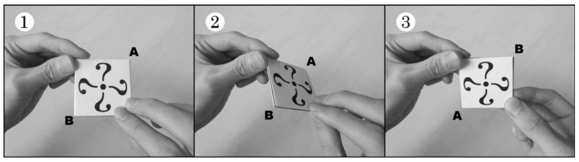


Fig. 6 An implementation of the diagonal flip operation τ.

Definition 4: Let $\rho \in \Sigma_k$ be a permutation, called a rotation operation, such that $\rho(x_0, \cdots, x_{k-1}) = (\overline{x_0}, \cdots, \overline{x_{k-1}})$. A *rotation shuffle* $\mathsf{Rotation}(\cdot)$ is defined as a shuffle based on ρ, i.e., $\mathsf{Rotation}(x_0, \cdots, x_{k-1}) = \rho^r(x_0, \cdots, x_{k-1})$ where r is uniformly chosen from $\{0, 1\}$, and it results in one of the two possibilities

$$\mathsf{Rotation}(x_0, \cdots, x_{k-1}) = \begin{cases} (x_0, \cdots, x_{k-1}) \\ (\overline{x_0}, \cdots, \overline{x_{k-1}}) \end{cases}$$

with probability exactly 1/2.

The rotation operation ρ for $(x_0, \cdots, x_{k-1})$ is physically implemented by stacking the k cards (using a cover; see Sect. 2.1) and then rotating them 90° (Fig. 5).

Next, we define a diagonal flip shuffle that is the main technical contribution in this paper:

Definition 5: Let $\tau \in \Sigma_k$ be a permutation, called a diagonal flip operation, such that $\tau(x_0, x_1, \cdots, x_{k-1}) = (\overline{x_{k-1}}, \cdots, \overline{x_1}, \overline{x_0})$. A *diagonal flip shuffle* $\mathsf{DiagFlip}(\cdot)$ is defined as a shuffle based on τ, i.e., $\mathsf{DiagFlip}(x_0, \cdots, x_{k-1}) = \tau^r(x_0, \cdots, x_{k-1})$ where r is uniformly chosen from $\{0, 1\}$, and it results in one of the two possibilities

$$\mathsf{DiagFlip}(x_0, \cdots, x_{k-1}) = \begin{cases} (x_0, x_1, \cdots, x_{k-1}) \\ (\overline{x_{k-1}}, \cdots, \overline{x_1}, \overline{x_0}) \end{cases}$$

with probability exactly 1/2.

The diagonal flip operation τ for $(x_0, \cdots, x_{k-1})$ is physically implemented by stacking the k cards (using a cover; see Sect. 2.1) and then flipping along the diagonal axis (Fig. 6).

3. Multi-Party Computation Using Polarizing Cards

In this section, we define the model of our polarizing-card-based protocols, and the security of our model.

3.1 Model

Now we are ready to define the notion of MPC protocols for polarizing cards. In contrast to standard MPC protocols, the number of parties is not essential in our protocols, as in most previous card-based protocols, since our protocols do not require private memories for parties, i.e., each of parties does not have any private information except his input information. Thus, our definition of a protocol is simply as follows.

Definition 6 (Protocol): A k-card protocol Π is specified by a transition table with multiple rows where each row of the transition table contains a symbol corresponding to one of the following five operations:

- $\langle \mathsf{Input} \rangle$: The protocol starts from here. This symbol only appears in the 0th row. Let s_0 be the initial sequence of k cards which encodes the input to the protocol[†] and contains any additional cards required by the protocol. Set the current sequence to s_0, and go to the next row.
- $\langle \mathsf{Permutation}, \sigma \rangle$ where $\sigma \in \Sigma_k$: Apply the permutation σ to the current sequence, and go to the next row.
- $\langle \mathsf{Superimposing}, (i, j), ind_0, ind_1 \rangle$ where $i, j \in \{0, \cdots, k-1\}$, $i \neq j$, and $ind_0 \neq ind_1$: Let γ be the result of superimposing the i-th and the j-th cards. If γ is *white*, then go to the ind_0-th row, otherwise go to the ind_1-th row.
- $\langle \mathsf{Shuffle}, \sigma \rangle$ where $\sigma \in \Sigma_k$: Apply the shuffle based on σ to the current sequence, and go to the next row.
- $\langle \mathsf{Output}, S \rangle$ where $S \subset \{0, \cdots, k-1\}$: Set the final sequence to the current sequence. The protocol outputs the subsequence indexed by S, and terminates.

From now on, for the simplicity, we use $\langle \mathsf{Shuffle}, name, S \rangle$ where the name is a shuffle name (e.g. $\mathsf{Rotation}$ or $\mathsf{DiagFlip}$), and S is a subset of $\{0, \cdots, k-1\}$ that designates the cards which the shuffle should be applied to.

For a protocol Π, let $s_0(x_0, x_1, \cdots, x_{n-1})$ be the initial sequence corresponding to the input values[††] $(x_0, x_1, \cdots, x_{n-1}) \in \{0, 1\}^n$. We say that a protocol Π computes $f(x_0, x_1, \cdots, x_{n-1})$ correctly if the following holds: for all $x_0, x_1, \cdots, x_{n-1}$,

$$\Pi(s_0(x_0, x_1, \cdots, x_{n-1})) \in \mathsf{Enc}(f(x_0, x_1, \cdots, x_{n-1}))$$

where $\Pi(s_0(x_0, x_1, \cdots, x_{n-1}))$ denotes the output sequence.

3.2 Security

We assume that all of parties are *honest-but-curious*, i.e., any parties do not deviate from the protocol. Roughly speaking, security of a protocol is that any unbounded adversary can not learn the input information when he can see an execution of the protocol. However, due to the direction indistinguishability property, any adversary should be unable to

[†]The alignment of the encoded input values must be specified by the protocol.

[††]Note that a vector of the input values is not equal to the initial sequence in general. For example, our XOR protocol (see Sect. 4.2) takes as an initial sequence $s_0 = (c_0, c_0 \oplus x, c_1, c_1 \oplus y)$. In the case, the vector of the input values is (x, y).

obtain information regarding the direction of a sequence of cards. For this reason, what an adversary can learn essentially is results of superimposing.

For a protocol Π, its move depends only on the input values x and the random values generated in shuffles. We use Γ_Π to denote a random variable representing the values of superimposing; $\Gamma_\Pi = (\gamma_0, \cdots, \gamma_{m-1})$, where $\gamma_i \in \{white, black\}$, means that the i-th result of superimposing in the execution is γ_i.

Definition 7 (Security): For a protocol Π, let X be a random variable taking over the input values $(x_0, x_1, \ldots, x_{n-1})$. We say that Π is secure if the following holds:

$$H(X|\Gamma_\Pi) = H(X),$$

where $H(\cdot)$ denotes the *Shannon entropy*.

Corollary 1: Let Π be a protocol. If Γ_Π is independent from the input values $(x_0, x_1, \ldots, x_{n-1})$ (e.g. each γ_i is either constant or random), then the protocol Π is secure.

We formally define minimality *with respect to an encoding scheme* as follows.

Definition 8 (Minimality): Let Π be a k-card protocol that securely computes $f : \{0,1\}^n \to \{0,1\}^m$ based on an ℓ-card-per-bit encoding scheme. We say that Π is *minimal* if there is no $(k-1)$-card protocol based on an ℓ-card-per-bit encoding scheme that securely computes f.

4. COPY, XOR, and AND Protocols

In this section, we construct COPY/XOR/AND protocols, using four cards. These protocols are minimal constructions since it is impossible to reduce the number of cards in the two-card-per-bit encoding scheme.

4.1 COPY Protocol

We construct a two-card COPY protocol that takes a card x, and outputs two copies of x. Using the protocol twice, we can immediately obtain a four-card COPY protocol for commitments.

0. $\langle \mathsf{Input} \rangle$: Let s_0 be the initial sequence as follows:

$$s_0 = (x, c).$$

where c is an arbitrary polarizing card.

1. $\langle \mathsf{Shuffle}, \mathrm{Rotation}, \{1\} \rangle$: Apply a rotation shuffle to c:

$$s_1 = (x, \mathsf{Rotation}(c)) = (x, r)$$

where r is uniformly chosen from $\{0, 1\}$.

2. $\langle \mathsf{Superimposing}, (0,1), 4, 3 \rangle$: Let γ be the result of superimposing x and r. If γ is *white*, i.e., $x = r$, then go to the 4th row, otherwise, $\overline{x} = r$, go to the 3rd row. Here, the values of the current sequence s_2 is the following two possibility according to γ:

$$s_2 = \begin{cases} (x, x) & \text{if } \gamma \text{ is } \textit{white}. \\ (x, \overline{x}) & \text{if } \gamma \text{ is } \textit{black}. \end{cases}$$

3. $\langle \mathsf{Permutation}, \rho_1 \rangle$: ρ_1 is a fundamental rotation (See Definition 1). Rotate the right card by 90°:

$$s_3 = \rho_1(x, \overline{x}) = (x, x).$$

4. $\langle \mathsf{Output}, \{0, 1\} \rangle$: The current sequence is s_1 if γ is *white*, otherwise s_3. The protocol outputs the current sequence, and terminates.

Theorem 1: The above COPY protocol is secure.

proof. A superimposing operation appears in the 2nd row and it only reveals $x \oplus r$. This is uniformly random since r is uniformly chosen from $\{0, 1\}$. *Hence, from corollary 1, our COPY protocol is secure.* □

4.2 XOR Protocol

Given two inputs $(c_0, c_0 \oplus x) \in \mathsf{Enc}(x)$ and $(c_1, c_1 \oplus y) \in \mathsf{Enc}(y)$, our four-card XOR protocol proceeds as follows.

0. $\langle \mathsf{Input} \rangle$: Let s_0 be the initial sequence as follows:

$$s_0 = (c_0, c_0 \oplus x, c_1, c_1 \oplus y).$$

1. $\langle \mathsf{Shuffle}, \mathrm{Rotation}, \{0, 1\} \rangle$: Apply a rotation shuffle to $(c_0, c_0 \oplus x)$:

$$s_1 = (\mathsf{Rotation}(c_0, c_0 \oplus x), c_1, c_1 \oplus y) = (r, r \oplus x, c_1, c_1 \oplus y)$$

where r is uniformly chosen from $\{0, 1\}$.

2. $\langle \mathsf{Superimposing}, (0,2), 4, 3 \rangle$: Let γ be the result of superimposing r and c_1. If γ is *white*, i.e., $r = c_1$, then go to the 4th row, otherwise, i.e., $\overline{r} = c_1$, go to the 3rd row. Here, the values of the current sequence s_2 is the following two possibility according to γ:

$$s_2 = \begin{cases} (r, r \oplus x, r, r \oplus y) & \text{if } \gamma \text{ is } \textit{white}. \\ (r, r \oplus x, \overline{r}, \overline{r} \oplus y) & \text{if } \gamma \text{ is } \textit{black}. \end{cases}$$

3. $\langle \mathsf{Permutation}, \rho_2 \circ \rho_3 \rangle$: ρ_2 and ρ_3 are fundamental rotations (See Definition 1). Rotate $\overline{r}$ and $\overline{r} \oplus y$ by 90°:

$$s_3 = \rho_2 \circ \rho_3(s_2) = (r, r \oplus x, r, r \oplus y).$$

4. $\langle \mathsf{Output}, \{1, 3\} \rangle$: The current sequence is s_1 if γ is *white*, otherwise s_3. The protocol outputs the 1st and the 3rd cards of the current sequence (Note that the far-left one is the 0th card), and terminates.

Theorem 2: The above XOR protocol is secure.

proof. A superimposing operation appears in the 2nd row, and only reveals $r \oplus c_1$. This is uniformly random since r is uniformly chosen from $\{0, 1\}$. *Hence, from corollary 1, our XOR protocol is secure.* □

4.3 AND Protocol

Given two inputs $(c_0, c_0 \oplus x) \in \mathsf{Enc}(x)$ and $(c_1, c_1 \oplus y) \in$

$\mathsf{Enc}(y)$, our four-card AND protocol proceeds as follows.

0. $\langle\mathsf{Input}\rangle$: Let s_0 be the initial sequence as follows:

$$s_0 = (c_0, c_0 \oplus x, c_1, c_1 \oplus y).$$

1. Apply our XOR protocol to s_0. Let s_1 be the final sequence of the XOR protocol:

$$s_1 = (r, r \oplus x, r, r \oplus y)$$

where r is uniformly chosen from $\{0, 1\}$.

2. $\langle\mathsf{Permutation}, \sigma_2 \circ \rho_0\rangle$: ρ_0 is a fundamental rotation and σ_2 is an adjacent transposition. Rotate the 0th card r by 90° and interchange the 2nd and the 3rd cards:

$$s_2 = \sigma_2 \circ \rho_0(s_1) = (\overline{r}, r \oplus x, r \oplus y, r).$$

3. $\langle\mathsf{Shuffle}, \mathrm{DiagFlip}, \{0, 1, 2\}\rangle$: Apply a diagonal flip shuffle to $(\overline{r}, r \oplus x, r \oplus y)$:

$$\begin{aligned} s_3 &= (\mathsf{DiagFlip}(\overline{r}, r \oplus x, r \oplus y), r) \\ &= \begin{cases} (\overline{r}, r \oplus x, r \oplus y, r) & \text{(A)} \\ (\overline{r \oplus y}, \overline{r \oplus x}, r, r) & \text{(B).} \end{cases} \\ &= \begin{cases} (\overline{r} \oplus (x \wedge y), r, r \oplus (\overline{x} \wedge y), r) \\ \text{— when (A)}\wedge(x = 0) \text{ or (B)}\wedge(x = 1). \\ (\overline{r} \oplus (\overline{x} \wedge y), \overline{r}, r \oplus (x \wedge y), r) \\ \text{— when (A)}\wedge(x = 1) \text{ or (B)}\wedge(x = 0). \end{cases} \end{aligned}$$

where (A) and (B) are the events corresponding to the two different outcomes of the randomized shuffle.

4. $\langle\mathsf{Superimposing}, (1, 3), 5, 7\rangle$: Let γ be the result of superimposing the 1st card and the 3rd card. If γ is *white*, i.e., the 1st card is r, then go to the 5th row, otherwise, i.e., the 1st card is $\overline{r}$, go to the 7th row. Here, the values of the current sequence s_4 is the following two possibility according to γ:

$$s_4 = \begin{cases} (\overline{r} \oplus (x \wedge y), r, r \oplus (\overline{x} \wedge y), r) & \text{if } \gamma \text{ is } \textit{white}. \\ (\overline{r} \oplus (\overline{x} \wedge y), \overline{r}, r \oplus (x \wedge y), r) & \text{if } \gamma \text{ is } \textit{black}. \end{cases}$$

5. $\langle\mathsf{Permutation}, \rho_0\rangle$: Rotate the 0th card by 90°:

$$s_5 = \rho_0(s_4) = (r \oplus (x \wedge y), r, r \oplus (\overline{x} \wedge y), r).$$

6. $\langle\mathsf{Output}, \{0, 3\}\rangle$: This is the output phase when γ is *white*. Thus, the current sequence is $s_5 = (r \oplus (x \wedge y), r, r \oplus (\overline{x} \wedge y), r)$. The protocol outputs the 0th and the 3rd cards of the current sequence, i.e., $(r \oplus (x \wedge y), r)$, and terminates.
7. $\langle\mathsf{Output}, \{2, 3\}\rangle$: This is the output phase when γ is *black*. Thus, the current sequence is $s_4 = (\overline{r} \oplus (\overline{x} \wedge y), \overline{r}, r \oplus (x \wedge y), r)$. The protocol outputs the 2nd and the 3rd cards of the current sequence, i.e., $(r \oplus (x \wedge y), r)$, and terminates.

Theorem 3: The above AND protocol is secure.

proof. A superimposing operations appear in the 1st row and the 4th row. The result in the 1st row is uniformly random (see the security of XOR protocol 4.2). The superimposing in the 4th row results in $(r \oplus x) \oplus r$ or $(\overline{r \oplus x}) \oplus r$ with probability exactly 1/2. This is uniformly random. Hence, from corollary 1, our AND protocol is secure. □

5. Applications

In this section, we construct a minimal input-preserving XOR protocol and a minimal input-preserving AND protocol, each of which uses four cards. Based on these protocols and applying the techniques from [5], [12], we obtain minimal half-adder/full-adder/voting protocols.

(1) Input-Preserving XOR Protocol

This protocol is immediately obtained from our XOR and COPY protocols. Let $(r, r \oplus x, r, r \oplus y)$ be the final sequence of the XOR protocol, where x and y are input bits and r is a random bit. Then, by applying our COPY protocol, we obtain the output $(r, r \oplus x, r \oplus x, r \oplus y)$, where $(r, r \oplus x)$ is a commitment to x, and $(r \oplus x, r \oplus y)$ is a commitment to $x \oplus y$.

(2) Input-Preserving AND Protocol

We design a minimal four-card input-preserving AND protocol. It takes two commitments to x, y as inputs, and outputs two commitments to $x \wedge y, x$ as follows. Let $s_0 = (c_0, c_0 \oplus x, c_1, c_1 \oplus y)$ be the initial sequence. Apply our AND protocol to $(c_1, c_1 \oplus y, c_0, c_0 \oplus x)$, and permute the final sequence of the AND protocol, we obtain the sequence $s_1 = (r, r \oplus (y \wedge x), r, r \oplus (\overline{y} \wedge x))$. Applying our COPY protocol to s_1, we can obtain the sequence $s_2 = (r, r \oplus (y \wedge x), r \oplus (y \wedge x), r \oplus (\overline{y} \wedge x))$. This can be simplified into $(c_2, c_2 \oplus (x \wedge y), c_3, c_3 \oplus x)$.

(3) Half-Adder and Full-Adder Protocols

Based on our input-preserving XOR/ AND protocols, we obtain the construction of a four-card half-adder protocol and a six-card full-adder protocol. Note that they are the minimal constructions since four/six cards are required to represent the output commitments of half-adder/full-adder protocols. These constructions are the same as [5], [12], while the half-adder and full-adder protocols from [5], [12] require six cards and ten cards, respectively.

(4) Voting Protocol

Secure voting is one of the most suitable applications for card-based protocols. Similarly to [5], using half-adder and full-adder protocols, we can construct a secure voting protocol with two candidates. Our voting protocol for n parties requires $2\lceil\log n\rceil + 2$ cards while the protocol from [5] requires $2\lceil\log n\rceil + 6$ cards. Here, the number of cards is minimal since a voting protocol takes as inputs $2\lceil\log n\rceil$ cards for a voting result of $n - 1$ parties and 2 cards for an input of n-th party.

6. One-Card-Per-Bit Encoding Scheme

In this section, we construct elementary protocols using an encoding which encodes one bit by one card, referred to as *one-card-per-bit encoding scheme*. Using these protocols in combination with a two-card-per-bit encoding scheme, we can compute any boolean function with a minimum number of cards. We also discuss one-card-per-bit encoding schemes and tradeoffs they provide.

6.1 Constructions

To construct a one-card-per-bit encoding scheme, we require a material with polarizing property that specifies a standard direction. This card is placed in the public space, and is referred to as a *common reference polarizer* crp. (The common reference polarizer is not required to have rotational symmetries since it is not used in the protocol except in a superimposing operation.) A value of a card x is determined by superimposing x and crp, i.e., $x = 0$ if the superimposing result is white, $x = 1$ if the superimposing result is black. We define a new operation $\mathsf{CheckValue}$ that takes a card x and outputs the value of the card x. This can be implemented by superimposing x and crp.

(1) **COPY Protocol**

COPY protocol for our one-card-per-bit encoding scheme is easily obtained from one presented in Sect. 4.1.

(2) **XOR Protocol**

0. $\langle\mathsf{Input}\rangle$: Let s_0 be the initial sequence as follows:
$$s_0 = (x, y).$$
1. $\langle\mathsf{Shuffle}, \text{Rotation}, \{0, 1\}\rangle$: Apply a rotation shuffle:
$$s_1 = (\mathsf{Rotation}(x, y)) = (r \oplus x, r \oplus y)$$
where r is uniformly chosen from $\{0, 1\}$.
2. $\langle\mathsf{CheckValue}, 0\rangle$: Let $\epsilon := r \oplus x$. If $\epsilon = 0$, then go to the 4th row, otherwise, i.e., $\epsilon = 1$, go to the 3rd row:
3. $\langle\mathsf{Permutation}, \rho_1\rangle$: ρ_1 is fundamental rotation (See Definition 1). Rotate $r \oplus y$ by 90°:
$$s_3 = \rho_1(s_1) = (r \oplus x, (r \oplus y) \oplus 1) = (r \oplus x, x \oplus y).$$
where the right side equality holds due to $\epsilon = r \oplus x = 1$.
4. $\langle\mathsf{Output}, \{1\}\rangle$: The current sequence is s_1 if $\epsilon = 0$, otherwise s_3. The protocol outputs the right card, and terminates.

Theorem 4: The above XOR protocol is secure.

proof. A superimposing operation only appears in the 2nd row as $\mathsf{CheckValue}$*, and reveals* $r \oplus x$*. This is uniformly random since r is uniformly chosen from* $\{0, 1\}$*. Hence, from corollary 1, our XOR protocol is secure.* □

(3) **AND Protocol**

Given two values represented by cards x, y, in our one-bit-per-card encoding scheme, our three-card AND protocol proceeds as follows.

0. $\langle\mathsf{Input}\rangle$: Let s_0 be the initial sequence as follows:
$$s_0 = (1, x, y).$$
where 1 is an additional cards whose value is 1.
1. $\langle\mathsf{Shuffle}, \text{DiagFlip}, \{0, 1, 2\}\rangle$: Apply a diagonal flip shuffle to $(1, x, y)$:
$$s_2 = \mathsf{DiagFlip}(1, x, y) = \begin{cases} (1, x, y) & \text{(A)} \\ (\overline{y}, \overline{x}, 0) & \text{(B)} \end{cases}.$$
where (A) and (B) are the events corresponding to the two different outcomes of the randomized shuffle.
2. $\langle\mathsf{CheckValue}, 1\rangle$: Let ϵ be the value of the center card. If $\epsilon = 0$, then go to the 3rd row, otherwise, i.e., $\epsilon = 1$, go to the 5th row.
3. $\langle\mathsf{Permutation}, \rho_0\rangle$: Rotate the 0th card by 90°.
$$s_3 = \begin{cases} (0, x, y) & \text{in the case of (A).} \\ (y, \overline{x}, 0) & \text{in the case of (B).} \end{cases}$$
4. $\langle\mathsf{Output}, \{0\}\rangle$: This is the output phase when $\epsilon = 0$. The current sequence is s_3. The protocol outputs the left (0th) card, and terminates.
5. $\langle\mathsf{Output}, \{2\}\rangle$: This is the output phase when $\epsilon = 1$. The current sequence is s_2. The protocol outputs the right (2nd) card, and terminates.

Theorem 5: The above AND protocol is secure.

proof. A superimposing operation only appears in the 2nd row as $\mathsf{CheckValue}$*. This results in* x *or* $\overline{x}$ *with probability exactly 1/2. Hence, from corollary 1, our AND protocol is secure.* □

Here, in the output sequence, the value of the remaining card (opposite to the output card) is $\overline{x} \wedge y$. Applying the COPY protocol to the output card and the center card, we can obtain two copies of $x \wedge y$. Thus, y can be computed by $(x \wedge y) \oplus (\overline{x} \wedge y)$. This modification allows the construction of an input-preserving AND protocol using three cards.

6.2 Application to Two-Cards-Per-Bit Protocols

We will now show that, utilizing the protocols constructed for the one-card-per-bit encoding scheme, we can construct a protocol for an arbitrary n-variable boolean function based on a two-card-per-bit encoding scheme, using only $2n + 2$ cards. Our starting point is a protocol for an arbitrary boolean function proposed by Nishida et al. [12]. They showed (a simplified version) of the following theorem.

Theorem 6 (Theorem 6 in [12] (generalized version)): Let f be an n-variable boolean function. If there exist a k_{C}-card COPY, a k_{A}-card input-preserving AND, and a k_{X}-card XOR protocols in an ℓ-card-per-bit encoding scheme, then given commitments to $x_0, x_1, \cdots, x_{n-1}$ based on the ℓ-card-per-bit encoding scheme, we can securely compute commitments to $x_0, x_1, \cdots, x_{n-1}$ and $f(x_0, x_1, \cdots, x_{n-1})$, using

$\max(k_\mathsf{C}, k_\mathsf{A}, k_\mathsf{X})$ additional cards.

In the case of the previously used two-card-per-bit encoding scheme ($\boxed{\clubsuit}$, $\boxed{\heartsuit}$), any function can be securely computed with six additional cards since there exist a six-card COPY ($k_\mathsf{C} = 6$), a six-card input-preserving AND protocol ($k_\mathsf{A} = 6$), and a four-card XOR protocol ($k_\mathsf{X} = 4$). In the case of our one-card-per-bit encoding scheme, two additional cards are required to compute any f since $k_\mathsf{C} = 2, k_\mathsf{A} = 3, k_\mathsf{X} = 2$ (note that the encoding scheme requires a common reference polarizer). Utilizing the one-card-per-bit protocols, we can securely compute an arbitrary boolean function $f : \{0, 1\}^n \to \{0, 1\}$ using a two-card-per-bit encoding scheme and only $2n+2$ cards, as follows. (Note that this is the minimum number of cards since the protocol preserves inputs commitments, and thus the number of output cards is $2n + 2$.) Given commitments to $x_0, \cdots, x_{n-1}$, generate $(r, r \oplus x_0), \cdots, (r, r \oplus x_{n-1})$ by applying a rotation shuffle and a superimposing operation. This can be regarded as n commitments based on one-card-per-bit encoding $(x_0, \cdots, x_{n-1})$, common reference polarizer r, and $n + 1$ free cards (the total number of cards is $2n + 2$). Using the one-card-per-bit protocol for f, it is possible to produce (one-card-per-bit) commitments to $x_0, x_1, \cdots, x_{n-1}$ and $f(x_0, \cdots, x_{n-1})$. Now, the number of remaining cards is $n + 1$ (including the common reference polarizer r). Applying the COPY protocol to the remaining cards, we can produce $(n + 1)$-copies of r, immediately obtain (two-card-per-bit) commitments to $x_0, x_1, \cdots, x_{n-1}$ and $f(x_0, \cdots, x_{n-1})$. This is a two-card-per-bit protocol for an arbitrary n-variable boolean function using only $2n + 2$ cards. (If the input commitments are not required to be preserved, $f(x_0, \cdots, x_{n-1})$ can be computed using only $2n$ cards. This gives the same constructions of our COPY/AND/XOR protocols in Sect. 4.)

6.3 Discussions

In this section, we constructed elementary protocols based on the one-card-per-bit encoding scheme. As a related work, Mizuki and Shizuya [8] proposed another one-card-per-bit encoding scheme using cards with a rotationally symmetric back side. The cards with a rotationally symmetric back side [8] enable the construction of a three-card COPY protocol, a two-card XOR protocol, and a three-card AND protocol. However, the shuffle used in the AND protocol is not known to have an easy physical implementation, i.e., it requires two operations, a rotation and a permutation, at the same time. In contrast to [8], the diagonal flip shuffle used in our one-card-per-bit AND protocol has an easy physical implementation (Sect. 2.2). Moreover, our one-card-per-bit COPY protocol in our scheme only uses two cards while the scheme [8] requires three cards. However, our scheme requires a common reference polarizer for the encoding. Therefore, there is a trade-off between the previous cards [8] and polarizing cards in one-card-per-bit encoding scheme.

Acknowledgment

The authors would like to thank members of the study group "Shin-Akarui-Angou-Benkyou-Kai" for the valuable discussions and helpful comments. This work was partially supported by JSPS KAKENHI Grant Numbers 26330001 and 26330151, Kurata Grant from The Kurata Memorial Hitachi Science and Technology Foundation, and JSPS A3 Foresight Program.

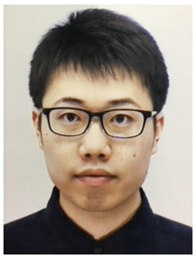
References

[1] B. den Boer, "More efficient match-making and satisfiability: The five card trick," Advances in Cryptology EUROCRYPT'89, Lecture Notes in Computer Science, vol.434, pp.208–217, Springer Berlin Heidelberg, 1990.
[2] C. Crépeau and J. Kilian, "Discreet solitary games," Advances in Cryptology, CRYPTO'93, Lecture Notes in Computer Science, vol.773, pp.319–330, Springer Berlin Heidelberg, 1994.
[3] R. Ishikawa, E. Chida, and T. Mizuki, "Efficient card-based protocols for generating a hidden random permutation without fixed points," Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, vol.9252, pp.215–226, Springer International Publishing, 2015.
[4] A. Koch, S. Walzer, and K. Härtel, "Card-based cryptographic protocols using a minimal number of cards," accepted in ASIACRYPT 2015, Cryptology ePrint Archive, Report 2015/865, 2015. http://eprint.iacr.org/, 2015.
[5] T. Mizuki, I.K. Asiedu, and H. Sone, "Voting with a logarithmic number of cards," Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, vol.7956, pp.162–173, Springer Berlin Heidelberg, 2013.
[6] T. Mizuki, M. Kumamoto, and H. Sone, "The five-card trick can be done with four cards," Advances in Cryptology, ASIACRYPT 2012, Lecture Notes in Computer Science, vol.7658, pp.598–606, Springer Berlin Heidelberg, 2012.
[7] T. Mizuki and H. Shizuya, "A formalization of card-based cryptographic protocols via abstract machine," Int. J. Inf. Secur., vol.13, no.1, pp.15–23, 2014.
[8] T. Mizuki and H. Shizuya, "Practical card-based cryptography," Fun with Algorithms, Lecture Notes in Computer Science, vol.8496, pp.313–324, Springer International Publishing, 2014.
[9] T. Mizuki and H. Sone, "Six-card secure AND and four-card secure XOR," Frontiers in Algorithmics, Lecture Notes in Computer Science, vol.5598, pp.358–369, Springer Berlin Heidelberg, 2009.
[10] T. Mizuki, F. Uchiike, and H. Sone, "Securely computing XOR with 10 cards," Australasian Journal of Combinatorics, vol.36, pp.279–293, 2006.
[11] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone, "Securely computing three-input functions with eight cards," IEICE Trans. Fundamentals, vol.E98-A, no.6, pp.1145–1152, June 2015.
[12] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone, "Card-based protocols for any boolean function," Theory and Applications of Models of Computation, Lecture Notes in Computer Science, vol.9076, pp.110–121, Springer International Publishing 2015.
[13] A. Nishimura, T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone, "Five-card secure computations using unequal division shuffle," Theory and Practice of Natural Computing, Lecture Notes in Computer Science, vol.9477, pp.109–120, Springer International Publishing, 2015.
[14] V. Niemi and A. Renvall, "Secure multiparty computations without computers," Theor. Comput. Sci., vol.191, no.1-2, pp.173–183, 1998.
[15] K. Shinagawa, T. Mizuki, J. Schuldt, K. Nuida, N. Kanayama,

T. Nishide, G. Hanaoka, and E. Okamoto, "Secure multi-party computation using polarizing cards," Advances in Information and Computer Security, Lecture Notes in Computer Science, vol.9241, pp.281–297, Springer International Publishing, 2015.

[16] K. Shinagawa, T. Mizuki, J.C.N. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto, "Multi-party computation with small shuffle complexity using regular polygon cards," Provable Security, Lecture Notes in Computer Science, vol.9451, pp.127–146, Springer International Publishing, 2015.

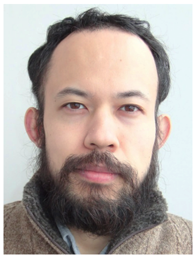
[17] A. Stiglic, "Computations with a deck of cards," Theor. Comput. Sci., vol.259, no.1-2, pp.671–678, 2001.



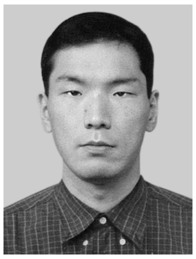
Kazumasa Shinagawa received his B.E. degree from University of Tsukuba in 2015. He is now a first grade master student of University of Tsukuba. He received SCIS Best Paper Award from IEICE in 2015.

Takaaki Mizuki received his B.E. degree in information engineering and his M.S. and Ph.D. degrees in information sciences from Tohoku University, Japan, in 1995, 1997 and 2000, respectively. He is currently an associate professor of the Cyberscience Center, Tohoku University. His research interests include cryptology and information security. He is a member of IEICE, IEEE, and IPSJ.

Jacob C. N. Schuldt obtained a B.Sc. degree and a M.Sc. degree (cand.scient) from The University of Copenhagen, and a Ph.D. degree from The University of Tokyo. He is currently a research scientist in the Advanced Cryptosystems Research Group, National Institute of Advanced Industrial Science and Technology (AIST), Japan. Before joining AIST, he held postdoctoral research positions at AIST and Royal Holloway, University of London.



Koji Nuida received the Ph.D. degree in Mathematical Science from The University of Tokyo, Japan, in 2006. From 2006, he had been working as a postdoctoral researcher, a researcher and currently a senior researcher at National Institute of Advanced Industrial Science and Technology (AIST), Japan. He is currently also receiving support as a Japan Science and Technology Agency (JST) PRESTO Researcher. His research interest is mainly in mathematics and mathematical cryptography.



Naoki Kanayama received his B.E., B.S., M.S. and D.S. degrees from Waseda University, Tokyo, Japan, in 1994, 1996, 1998 and 2003, respectively. In 2003–2006, he was a postdoctoral fellow of the Japan Society for the Promotion of Science. In 2006–2013, he was a research fellow at University of Tsukuba. He is an assistant professor at University of Tsukuba. Dr. Kanayama is a member of the Japan Society for Industrial and Applied Mathematics and of the Information Processing Society of Japan.

Takashi Nishide received B.S. degree from the University of Tokyo in 1997, M.S. degree from the University of Southern California in 2003, and Dr.E. degree from the University of Electro-Communications in 2008. From 1997 to 2009, he had worked at Hitachi Software Engineering Co., Ltd. developing security products. From 2009 to 2013, he had been an assistant professor at Kyushu University and from 2013 he is an associate professor at University of Tsukuba. His research is in the areas of cryptography and information security.

Goichiro Hanaoka graduated from the Department of Engineering, The University of Tokyo in 1997. Received Ph.D. degree from The University of Tokyo in 2002. Joined AIST in 2005. Currently, Leader, Advanced Cryptosystems Research Group, Information Technology Research Institute, AIST. Engages in the R&Ds for encryption and information security technologies including the efficient design and security evaluation of public key cryptosystem. Received the Wilkes Award (2007), British Computer Society; Best Paper Award (2008), The Institute of Electronics, Information and Communication Engineers; Innovative Paper Award (2012, 2014), Symposium on Cryptography and Information Security (SCIS); Award of Telecommunication Advancement Foundation (2005); 20th Aniversary Award (2005), SCIS; Best Paper Award (2006), SCIS; Encouragement Award (2000), Symposium on Information Theory and its Applications (SITA); and others.

Eiji Okamoto received his B.S., M.S. and Ph.D. degrees in electronics engineering from the Tokyo Institute of Technology in 1973, 1975 and 1978, respectively. He worked and studied communication theory and cryptography for NEC central research laboratories since 1978. In 1991 he became a professor at Japan Advanced Institute of Science and Technology, then at Toho University. Now he is a professor at Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests are cryptography and information security. He is members of IEEE and ACM.