Card-Based Protocols Using Regular Polygon Cards

著者	SHINAGAWA Kazumasa, MIZUKI Takaaki, SCHULDT Jacob C.N., NUIDA Koji, KANAYAMA Naoki, NISHIDE Takashi, HANAOKA Goichiro, OKAMOTO Eiji
journal or publication title	IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences
volume	E100.A
number	9
page range	1900-1909
year	2017-09
権利	(C) 2017 The Institute of Electronics, Information and Communication Engineers
URL	http://hdl.handle.net/2241/00150564

doi: 10.1587/transfun.E100.A.1900

PAPER *Special Section on Discrete Mathematics and Its Applications*

Card-Based Protocols Using Regular Polygon 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 Cryptographic protocols enable participating parties to compute any function of their inputs without leaking any information beyond the output. A card-based protocol is a cryptographic protocol implemented by physical cards. In this paper, for constructing protocols with small numbers of shuffles, we introduce a new type of cards, *regular polygon cards*, and a new protocol, *oblivious conversion*. Using our cards, we construct an addition protocol on non-binary inputs with only one shuffle and two cards. Furthermore, using our oblivious conversion protocol, we construct the first protocol for general functions in which the number of shuffles is linear in the number of inputs.
key words: *card-based protocol, regular polygon cards*

1. Introduction

1.1 Background

In 1989, den Boer [2] proposed a protocol called the *Five-Card Trick*, which can securely compute the AND function, using five cards that have two types of front sides (♣, ♡) and identical back sides (?). The feasibility of basing cryptographic protocols on this, i.e., *what functions can be securely computed by these cards*, was solved by the subsequent works [1], [9]. On the other hand, the efficiency, i.e., *how many cards and shuffles are sufficient to compute a function*, is still an important question.

In terms of the number of cards, Nishida et al. [11] showed that for any Boolean function $f : \{0, 1\}^n \to \{0, 1\}$, it is possible to construct a $(2n + 6)$-card protocol, using the elementary protocols proposed by Mizuki and Sone [9]. Since n-bit input uses $2n$ cards, their result showed that only six additional cards are sufficient to compute any function. However, it has remained an open problem to provide upper bounds on the number of *shuffles* required to compute any function.

1.2 Our Contribution

In this paper, we propose new techniques for constructing a card-based protocol with small number of shuffles. The first technique is to introduce a new type of cards, a *regular polygon card*. In contrast to all the previous works, our card can deal with multiple values naturally. This leads to a new type of protocols using only a small number of shuffles, which cannot be achieved using the previous cards. The second technique is an *oblivious conversion*, which is a new protocol. It is used to construct a protocol for general functions using only a small number of shuffles. The details of our contribution are follows.

Table 1 Comparison between our protocols and previous protocols.

	Card	# of shuffles	# of cards
∘ Addition and Subtraction over $\mathbb{Z}/m\mathbb{Z}$			
[4], [9] based	standard	$O(\log m)$	$O(\log m)$
Ours	m-sided	1	2
∘ Multiplication by $c \in \mathbb{Z}/m\mathbb{Z}$			
[4], [9] based	standard	$O(\log c \cdot \log m)$	$O(\log c \cdot \log m)$
Ours	m-sided	$\lceil \log_2 c \rceil + 1$	$\lceil \log_2 c \rceil + 2$
∘ Protocol for an arbitrary $f : (\mathbb{Z}/m\mathbb{Z})^n \to \mathbb{Z}/m\mathbb{Z}$			
[11] based	standard	$O(m^n \cdot \log m)$	$2((n + 1)\lceil \log_2 m \rceil + 2)$
Ours	m-sided	n	$m + n + m^n$
∘ Protocol for an arbitrary $f : (\mathbb{Z}/2\mathbb{Z})^n \to \mathbb{Z}/2\mathbb{Z}$			
[11]	standard	$O(2^n)$	$2(n + 3)$
Ours	standard	n	$2(n + 2^n)$

The regular m-sided polygon cards have $(360/m)^\circ$ rotational symmetry. Using the cards introduced by den Boer [2] (hereafter the standard cards), the previous addition protocols over $\mathbb{Z}/m\mathbb{Z}$ require that the numbers of shuffles and cards are proportional to $\log m$. On the other hand, using the regular m-sided polygon cards, we construct an addition protocol over $\mathbb{Z}/m\mathbb{Z}$ that requires one shuffle and two cards (Table 1). We also construct a multiplication protocol with $\lceil \log_2 c \rceil + 1$ shuffles, where c is the multiplication factor, while the previous binary protocol requires $O(\log c \cdot \log m)$ shuffles (Table 1).

Our oblivious conversion* is a protocol that takes an encoding of $a \in \mathbb{Z}/m\mathbb{Z}$ and a function f as inputs, and outputs an encoding of $f(a)$. Using it iteratively, we construct a protocol for any function $f(x_1, \cdots, x_n)$ with only $2n$ shuffles while it requires $O(2^n)$ number of cards (Table 1). We note that such a protocol can be implemented by both our polygon cards and the standard cards. This result is complementary to that of Nishida et al. [11]: they constructed a protocol for any function with only $2n + 6$ standard cards

Manuscript received September 26, 2016.
Manuscript revised January 31, 2017.
[†]The authors are with University of Tsukuba, Tsukuba-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 appeared at [16].
a) E-mail: shinagawa@cipher.risk.tsukuba.ac.jp
DOI: 10.1587/transfun.E100.A.1900

*Oblivious conversion is named after the oblivious transfer.

Table 2 Comparison of voting protocols for n voters.

	[4] (standard)	Ours (polygon)
# of candidates	2	ℓ
# of shuffles	$O(n \log n)$	$n+1$
# of cards	$2\lceil \log_2 n \rceil + 6$	$(n+2)\ell$

and $O(2^n)$ number of shuffles.

By designing a specific protocol in a careful way, we can achieve a protocol with both a small number of shuffles and cards. As an example, we construct a voting protocol. For n voters and ℓ candidates, our protocol uses $n+1$ shuffles and $(n+2)\ell$ cards (Table 2).

1.3 Related Works

In 1993, Crépeau and Kilian [1] achieved protocols implementing any function by constructing composable elementary protocols (COPY/XOR/AND). In 2009, Mizuki and Sone [9] constructed composable elementary protocols using fewer cards, by applying a new shuffle called a *random bisection cut*. Using these protocols, the number of shuffles needed to evaluate a function f is exactly the number of gates of f. Our construction (Sect. 4) improves the number of shuffles by the number of inputs, which is strictly smaller than the number of gates.

We note that almost all previous works [1]–[13], [17], [18] only consider binary inputs. Our polygon cards enable us to construct the first non-binary protocols.

2. Basic Notation

In this section, we introduce a *regular polygon card* and basic notations for describing card-based protocols.

2.1 Regular Polygon Cards

Let $m \geq 3$ be an integer. A regular m-sided polygon card is a card having a back side with $(360/m)°$ rotational symmetry and a front side with no rotational symmetry. For the sake of easy description, hereafter we use a concrete regular polygon card, a regular four-sided polygon card: its front side is [↑] and its back side is [■]. The elements of $\mathbb{Z}/4\mathbb{Z}$ (hereafter $\mathbb{Z}_4$) naturally correspond to rotations of a card as shown below.

[↑] = 0, [→] = 1, [↓] = 2, [←] = 3.

For $x \in \mathbb{Z}_4$, we use $[\![x]\!]$ to denote the back side of a card that corresponds to x. We also use x to denote not only an element in $\mathbb{Z}_4$ but also the front side card, as long as it is clear from the context. The important property is that $[\![0]\!], [\![1]\!], [\![2]\!]$ and $[\![3]\!]$ have the identical face [■].

Although a "two-sided polygon" makes little geometric sense, the card whose back side has a 180° rotationally symmetric pattern [8] can be regarded as a regular two-sided polygon card. Its front side is [↑] and its back side is [ ]. (Note that its shape is a rectangle instead of a square.) Clearly, the back side has 180° rotational symmetry.

We note that all of our protocols can be applied to m-sided polygon cards for any $m \geq 2$ while our descriptions use four-sided polygon cards.

2.2 Basic Definitions

We define basic definitions: *stack*, *sequence*, *top function*, *rotation function*, and *flip function*.

(1) Stack and Sequence

We first define a *stack* and a *stacking operation* "$\cdot$", recursively as follows.

- A card c is a stack.
- If d_1 and d_2 are stacks, then $d_1 \cdot d_2$ is a stack.

For example, for k cards $c_1, c_2, \cdots, c_k$, $d = c_1 \cdot c_2 \cdot \cdots \cdot c_k$ is a stack of k cards.

We next define a *sequence*, which is a line of stacks, recursively as follows.

- If d is a stack, (d) is a sequence.
- If $s = (d_1, \cdots, d_k)$ is a sequence and d is a stack, then $(d_1, \cdots, d_k, d)$ is a sequence.

(2) Top Function

Following the formalization [7], we define a *top function* top, which returns the visible face of a card, as follows. For a card with upward facing front side $x \in \{0, 1, 2, 3\}$, $\mathsf{top}(x) = x$ whereas $\mathsf{top}([\![x]\!]) = \perp$ (here, $\perp$ is a symbol meaning "back side"). For a stack $d = c_1 \cdots c_k$, $\mathsf{top}(d) = (\mathsf{top}(c_1))^k$, where superscript denotes the number of cards rather than exponentiation. This means that the visible face of the stack is the same as the visible face of the top card except the number of cards. For a sequence $s = (d_1, \cdots, d_k)$, $\mathsf{top}(s) = (\mathsf{top}(d_1), \cdots, \mathsf{top}(d_k))$.

Example 1: The following stacks s_1 and s_2 satisfy $\mathsf{top}(s_1) = \perp^2$ and $\mathsf{top}(s_2) = \perp^3$. The following sequence S_3 satisfies $\mathsf{top}(S_3) = (\perp, 2, \perp^2)$.

$s_1 = [\![0]\!] \cdot [\![1]\!] = \underbrace{[■]}_{[\![0]\!] \cdot [\![1]\!]}$. $s_2 = [\![0]\!] \cdot 1 \cdot [\![2]\!] = \underbrace{[■]}_{[\![0]\!] \cdot 1 \cdot [\![2]\!]}$.

$S_3 = ([\![0]\!], 2, [\![2]\!] \cdot 3) = \Big(\underbrace{[■]}_{[\![0]\!]}, [↓], \underbrace{[■]}_{[\![2]\!] \cdot 3} \Big)$.

(3) Rotation Function

We define a rotation function rot, which returns a card rotated by a clockwise 90° rotation, as follows. For a card with upward facing front side $x \in \{0, 1, 2, 3\}$, $\mathsf{rot}(x) = x+1 \bmod 4$ whereas $\mathsf{rot}([\![x]\!]) = [\![x-1 \bmod 4]\!]$. For a stack $d = c_1 \cdots c_k$, $\mathsf{rot}(d) = \mathsf{rot}(c_1) \cdots \mathsf{rot}(c_k)$. For a sequence $s = (d_1, \cdots, d_k)$, $\mathsf{rot}(s) = (\mathsf{rot}(d_1), \cdots, \mathsf{rot}(d_k))$.

Example 2:

$\mathsf{rot}(0) = \mathsf{rot}([↑]) = [→] = 1$.

$$\mathsf{rot}(\llbracket 0 \rrbracket) = \mathsf{rot}\Big(\underbrace{\boxed{\blacksquare}}_{\llbracket 0 \rrbracket}\Big) = \underbrace{\boxed{\blacksquare}}_{\llbracket 3 \rrbracket} = \llbracket 3 \rrbracket.$$

$$\mathsf{rot}(\llbracket 0 \rrbracket \cdot 0) = \mathsf{rot}\Big(\underbrace{\boxed{\blacksquare}}_{\llbracket 0 \rrbracket \cdot 0}\Big) = \underbrace{\boxed{\blacksquare}}_{\llbracket 3 \rrbracket \cdot 1} = \llbracket 3 \rrbracket \cdot 1.$$

(4) Flip Function

We define a flip function flip, which returns the flipped cards, as follows. For a card with upward facing front side $x \in \{0, 1, 2, 3\}$, $\mathsf{flip}(x) = \llbracket x \rrbracket$ whereas $\mathsf{flip}(\llbracket x \rrbracket) = x$. For a stack $d = c_1 \cdot c_2 \cdots c_{k-1} \cdot c_k$, $\mathsf{flip}(d) = \mathsf{flip}(c_k) \cdot \mathsf{flip}(c_{k-1}) \cdots \mathsf{flip}(c_2) \cdot \mathsf{flip}(c_1)$. For a sequence $s = (d_1, \cdots, d_k)$, $\mathsf{flip}(s) = (\mathsf{flip}(d_1), \cdots, \mathsf{flip}(d_k))$.

Example 3:

$$\mathsf{flip}(0) = \mathsf{flip}\Big(\boxed{\uparrow}\Big) = \underbrace{\boxed{\blacksquare}}_{\llbracket 0 \rrbracket} = \llbracket 0 \rrbracket.$$

$$\mathsf{flip}(\llbracket 0 \rrbracket \cdot \llbracket 1 \rrbracket) = \mathsf{flip}\Big(\underbrace{\boxed{\blacksquare}}_{\llbracket 0 \rrbracket \cdot \llbracket 1 \rrbracket}\Big) = \underbrace{\boxed{\rightarrow}}_{1 \cdot 0} = 1 \cdot 0.$$

2.3 Operations

(1) Basic Operations on a Sequence

Let $s = (d_1, \cdots, d_k)$ be a sequence. We define the following operations for s.

Transposition: For any $1 \leq i < j \leq k$, a *transposition operation* (i, j) for s returns the following sequence

$$(d_1, \cdots, d_{i-1}, d_j, d_{i+1}, \cdots, d_{j-1}, d_i, d_{j+1}, \cdots, d_k).$$

Since every permutation can be represented by transpositions, we can rearrange a sequence arbitrarily.

Rotation: For any $1 \leq i \leq k$, a *rotation operation* of the i-th stack for s returns the following sequence

$$(d_1, \cdots, d_{i-1}, \mathsf{rot}(d_i), d_{i+1}, \cdots, d_k).$$

Flip: For any $1 \leq i \leq k$, a *flip operation* of the i-th stack for s returns the following sequence

$$(d_1, \cdots, d_{i-1}, \mathsf{flip}(d), d_{i+1}, \cdots, d_k).$$

We call a flip operation *open* when the stack is a stacking of face-down cards.

Composition/Decomposition: For any $1 \leq i < j \leq k$, a *composition operation* of the i-th stack and the j-th stack for s returns the following sequence

$$(d_1, \cdots, d_{i-1}, d_i \cdot d_j, d_{i+1}, \cdots, d_{j-1}, d_{j+1}, \cdots, d_k).$$

If the i-th stack is $d_i = d \cdot c$, where d is a stack and c is a card, a *decomposition operation* of the i-th stack for s returns the following sequence

$$(d_1, \cdots, d_{i-1}, d, c, d_{i+1}, \cdots, d_k).$$

Composition/Decomposition with Flip: For any $1 \leq i < j \leq k$, a *composition operation with flip* of the i-th stack and the j-th stack for s returns the following sequence

$$(d_1, \cdots, d_{i-1}, d_i \cdot \mathsf{flip}(d_j), d_{i+1}, \cdots, d_{j-1}, d_{j+1}, \cdots, d_k).$$

We note that this operation can be done without revealing $\mathsf{face}(\mathsf{flip}(d_j))$ by utilizing a non-transparent cover to mask $\mathsf{face}(\mathsf{flip}(d_j))$. If the i-th stack is $d_i = c \cdot d$, where c is a card and d is a stack, a *decomposition operation with flip* of the i-th stack for s returns the following sequence

$$(d_1, \cdots, d_{i-1}, c, \mathsf{flip}(d), d_{i+1}, \cdots, d_k).$$

Similarly, this can be done without revealing $\mathsf{face}(d)$.

Insert/Delete An *insert operation* for s returns the following sequence

$$(d_1, \cdots, d_{k-1}, d_k, 0).$$

A *delete operation* for s returns the following sequence

$$(d_1, \cdots, d_{k-1}).$$

(2) Cyclic Shuffle

A cyclic shuffle (which is denoted by $\langle \cdot \rangle$)

$$\Big\langle \overset{1}{\boxed{\blacksquare}}\overset{2}{\boxed{\blacksquare}}\overset{3}{\boxed{\blacksquare}}\overset{4}{\boxed{\blacksquare}} \Big\rangle$$

results in one of the the following sequences

$$\overset{1}{\boxed{\blacksquare}}\overset{2}{\boxed{\blacksquare}}\overset{3}{\boxed{\blacksquare}}\overset{4}{\boxed{\blacksquare}}, \quad \overset{2}{\boxed{\blacksquare}}\overset{3}{\boxed{\blacksquare}}\overset{4}{\boxed{\blacksquare}}\overset{1}{\boxed{\blacksquare}}, \quad \overset{3}{\boxed{\blacksquare}}\overset{4}{\boxed{\blacksquare}}\overset{1}{\boxed{\blacksquare}}\overset{2}{\boxed{\blacksquare}}, \quad \overset{4}{\boxed{\blacksquare}}\overset{1}{\boxed{\blacksquare}}\overset{2}{\boxed{\blacksquare}}\overset{3}{\boxed{\blacksquare}}$$

each occurring with probability 1/4. In general, a cyclic shuffle takes a sequence $(s_1, s_2, \cdots, s_k)$ such that $\mathsf{top}(s_i) = \perp^{\ell_i}$ for some integer ℓ_i, and outputs one of the following sequences

$$\begin{cases} (s_1, s_2, s_3, \cdots, s_{k-1}, s_k) \\ (s_2, s_3, s_4, \cdots, , s_k, s_1) \\ \qquad \vdots \\ (s_k, s_1, s_2, \cdots, s_{k-2}, s_{k-1}) \end{cases}$$

each occurring with probability $1/k$.

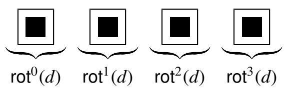
We say that a cyclic shuffle is an *equal shuffle* if $\mathsf{top}(s_1) = \mathsf{top}(s_2) = \cdots = \mathsf{top}(s_k)$. In this paper, we use only equal shuffles and rotation shuffles defined later. Recently, Nishimura et al. [13] showed that an unequal shuffle, which is not an equal shuffle, can be securely implemented by using a special type of boxes.

(3) Rotation Shuffle

For a stack d, a rotation shuffle (which is denoted by $(\cdot)$)

$$\Big(\underbrace{\boxed{\blacksquare}}_{d}\Big)$$

results in one of the four stacks



each occurring with probability 1/4. For example, for $d = [\![a]\!] \cdot [\![b]\!]$, a rotation shuffle results in one of the followings.

$$\begin{cases} [\![a]\!] \cdot [\![b]\!] \\ [\![a-1]\!] \cdot [\![b-1]\!] \\ [\![a-2]\!] \cdot [\![b-2]\!] \\ [\![a-3]\!] \cdot [\![b-3]\!] \end{cases}$$

On the other hand, for $d = [\![a]\!] \cdot b$, a rotation shuffle results in one of the followings.

$$\begin{cases} [\![a]\!] \cdot b \\ [\![a-1]\!] \cdot (b+1) \\ [\![a-2]\!] \cdot (b+2) \\ [\![a-3]\!] \cdot (b+3) \end{cases}$$

It plays an important role in designing our addition protocol (Sect. 3.1).

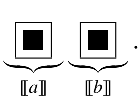
2.4 Security

Let Π be a protocol. Let $(\Gamma_0, \Gamma_1, \cdots, \Gamma_t)$ be a *history* of sequences in a protocol run, i.e., Γ_0 is an initial sequence determined by inputs, Γ_{i+1} arises from Γ_i by a physical operation (e.g. shuffle, rearrangement, open†), and Γ_t is a final sequence. Now we define a *visible sequence trace* by $(\mathsf{top}(\Gamma_0), \mathsf{top}(\Gamma_1), \cdots, \mathsf{top}(\Gamma_t))$. We say that Π is secure if a random variable of the visible sequence trace and a random variable of inputs are independent.

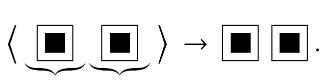
Definition 1 (Security): *Let Π be a protocol. Let V be a random variable of the visible sequence of Π and let U be the set of inputs of Π. We say that Π is secure if for any input distribution X on U, X and V are independet.*

Example 4: See the following (meaningless) protocol $\Pi_{\rm ex}$.

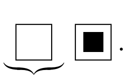
1. Place the two cards according to $a, b \in \{0, 1, 2, 3\}$:



2. Apply a cyclic shuffle:



3. Open the left-side card:



4. Output the right-side card.

The history of sequences in a protocol run, when the cyclic shuffle exchanges the two cards, is the following.

$$(\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3) = (([\![a]\!], [\![b]\!]), ([\![b]\!], [\![a]\!]), (b, [\![a]\!]), [\![a]\!]).$$

The random variable of the visible sequence of $\Pi_{\rm ex}$ is

$$V = ((\bot, \bot), (\bot, \bot), (\epsilon, \bot), \bot).$$

where ϵ is a random variable on $\{a, b\}$. The set of inputs U of the above protocol is as below.

$$U = \{(a, b) \mid 0 \leq a, b \leq 3\}.$$

$\Pi_{\rm ex}$ is *not* secure since ϵ depends on the inputs (a, b).

3. Addition Protocol

In this section, we construct an addition, a subtraction and a copy protocols, which use only a rotation shuffle. We also construct a c-multiplication protocol for any $c \in \mathbb{Z}_m$, which takes $[\![a]\!]$ and outputs $[\![ca]\!]$. It uses $(\lceil \log_2 c \rceil + 1)$ shuffles and $(\lceil \log_2 c \rceil + 2)$ cards.

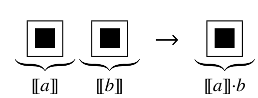
3.1 Addition Protocol

Our addition protocol takes $[\![a]\!]$ and $[\![b]\!]$ as inputs, and outputs $[\![a + b \bmod 4]\!]$. One can see the demonstration movie [15].

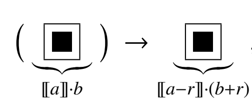
Protocol 1 (Addition Protocol):

- Input: $([\![a]\!], [\![b]\!])$.
- Output: $[\![a + b \bmod 4]\!]$.

1. Apply a composition with flip:

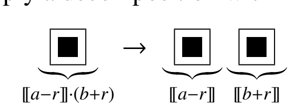


2. Apply a rotation shuffle:

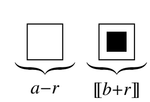


where r is a random integer with $0 \leq r \leq 3$.

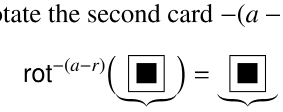
3. Apply a decomposition with flip to the stack:



4. Open the left-side card $[\![a - r]\!]$:



5. Rotate the second card $-(a - r)$ times and output it:



†We call by *open* an operation which turns over a back side card.

Theorem 1: *The above protocol is secure. It uses one shuffle and two cards.*

Proof. We prove the security of the above protocol, which uses one shuffle and two cards. Let A (or B) be a random variable of the first input (second input, respectively). Let $X = (A, B)$ be a random variable of the inputs. Let R be a random variable of the randomness used in the rotation shuffle. The random variable of the visible sequence V is

$$V = ((\bot, \bot), (\bot^2), (\bot^2), (\bot, \bot), (E, \bot), (\bot))$$

where $E = A - R \bmod 4$. E and A are independent since $\Pr[E = \epsilon \mid A = a] = 1/4$ and $\Pr[E = \epsilon] = 1/4$ for any $a, \epsilon \in \{0, 1, 2, 3\}$. Therefore, V and X are independent since E and X are also independent and V is just derived from E. Thus, the above protocol is secure. □

Corollary 1: *There is a secure protocol that takes as inputs* $[\![a]\!]$ *and* $([\![b_1]\!], \cdots, [\![b_k]\!])$*, and outputs* $([\![a + b_1]\!], \cdots, [\![a + b_k]\!])$ *with one shuffle and* $k + 1$ *cards. Especially, there is a secure protocol that takes as inputs* $[\![a]\!]$*, and outputs* k *copies of* $[\![a]\!]$ *with one shuffle and* $k+1$ *cards for any* $k \in \mathbb{N}$.

Proof. By replacing the stack $[\![a]\!] \cdot b$ with a stack $[\![a]\!] \cdot b_1 \cdot b_2 \cdot \cdots \cdot b_k$, we have a multiple addition protocol. This protocol uses one shuffle and $k + 1$ cards, and its security is proven in the same way as above. Applying the multiple addition protocol to the inputs $b_1 = b_2 = \cdots = b_k = 0$, we have a copy protocol that outputs k copies of $[\![a]\!]$. □

Subtraction is also possible using the same idea of the addition protocol. The differences are: use a stack $[\![a]\!] \cdot [\![b]\!]$ instead of $[\![a]\!] \cdot b$ and rotate with inverse direction in the last step. We omit the security proof since it is almost identical to the proof for the addition protocol.

Corollary 2: *There is a secure protocol that takes as inputs* $[\![a]\!]$ *and* $[\![b]\!]$*, and outputs* $[\![b - a]\!]$ *with one shuffle and two cards.*

3.2 Multiplication Protocol

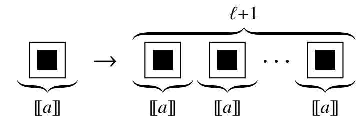
In this section, we construct a c-multiplication protocol for any public value $c \in \mathbb{Z}_m$, that takes $[\![a]\!]$ and outputs $[\![ca]\!]$. Trivially, such a computation can be done by using our addition protocol c times. On the other hand, it is well known that the number of additions can be reduced to $O(\log_2 c)$ (*binary method*). In this section, we design a multiplication protocol in a careful way and show that $(\lceil \log_2 c \rceil + 1)$ shuffles are sufficient to compute the multiplication $[\![ca]\!]$ from $[\![a]\!]$.

Protocol 2 (c-Multiplication Protocol):

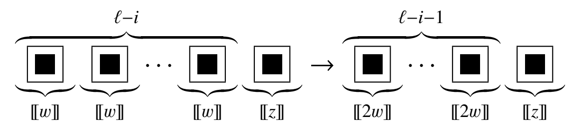
- Input: $[\![a]\!]$.
- Output: $[\![ca]\!]$.

Let $\ell = \lceil \log_2 c \rceil$ and $c - 1 = \sum_{j=0}^{\ell-1} 2^j \cdot b_j$ where $b_j \in \{0, 1\}$.

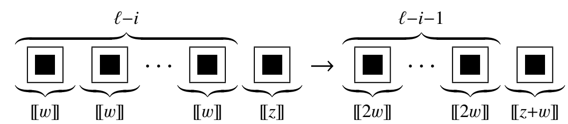
1. Invoke our $(\ell + 1)$-copy protocol to $[\![a]\!]$:



2. Let $W \leftarrow (\underbrace{[\![a]\!], \cdots, [\![a]\!]}_{\ell+1})$. For $i = 0, 1, \cdots, \ell - 1$, repeat the following.
   a. Let $W = (\overbrace{[\![w]\!], \cdots, [\![w]\!]}^{\ell-i}, [\![z]\!])$. (Note that $w = 2^i a$ and $z = (\sum_{j=0}^{i-1} 2^j b_j + 1)a$.)
   b. If $b_i = 0$, apply a multiple addition protocol to W except for $[\![z]\!]$:



   c. If $b_i = 1$, apply a multiple addition protocol to W:



   d. Update W to the current sequence. Note that the length of W has now decreased by one.
3. W is now just the rightmost card $[\![z]\!]$, where $z = (\sum_{j=0}^{\ell-1} 2^j b_j + 1)a = ca$. Output the card $[\![z]\!]$.

Theorem 2: *The above protocol is secure. It uses* $\lceil \log_2 c \rceil + 1$ *shuffles and* $\lceil \log_2 c \rceil + 2$ *cards.*

Proof. Let $\ell = \lceil \log_2 c \rceil$. Let A be a random variable of the input, and let V be a random variable of the visible sequence. Let E be a random variable of the opened value in copy protocol invoked in Step 1, and let E_i ($i \in \{1, \cdots, \ell\}$) be a random variable of the opened value in addition protocol invoked in the $(i-1)$-th iteration of Step 2. As mentioned in the proof of Theorem 1, A and E_i are independent. Moreover, A and $(E_0, E_1, \cdots, E_\ell)$ are also independent since each E_i is derived from each shuffle. Thus, A and V are independent since V essentially consists of $E_0, E_1, \cdots, E_\ell$. Therefore, it is secure. □

Example 5: Let $c = 6$. Here, $\ell = \lceil \log_2 c \rceil = 3$ and $5 = \sum_{i=0}^{2} 2^i b_i = 2^0 \cdot 1 + 2^1 \cdot 0 + 2^2 \cdot 1$. The execution process of c-multiplication protocol is as follows.

1. $[\![a]\!] \xrightarrow{\text{Copy 4}} ([\![a]\!], [\![a]\!], [\![a]\!], [\![a]\!])$.
2. $([\![a]\!], [\![a]\!], [\![a]\!], [\![a]\!]) \xrightarrow{\text{Add}} ([\![2a]\!], [\![2a]\!], [\![2a]\!])$.
3. $([\![2a]\!], [\![2a]\!], [\![2a]\!]) \xrightarrow{\text{Add}} ([\![4a]\!], [\![2a]\!])$.
4. $([\![4a]\!], [\![2a]\!]) \xrightarrow{\text{Add}} [\![6a]\!]$.

4. Oblivious Conversion

In this section, we introduce a new protocol, *oblivious conversion*, that enables secure computation for general functions with a small number of shuffles.

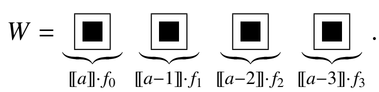
4.1 Oblivious Conversion

The oblivious conversion protocol takes as input a value $[\![a]\!]$, $a \in \mathbb{Z}_m$, and an encoding of a function f using a sequence of stacks $(f_1, \cdots, f_{m-1})$ where $\mathsf{top}(f_i) = \perp^k$ for some integer k. Each stack f_i is regarded as an encoding of $f(i)$. The output of the protocol will be f_a, which corresponds to an encoding of $f(a)$. For simplicity, we will set $m = 4$ in the following description. One can see the demonstration movie [14].

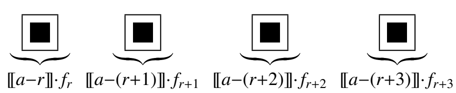
Protocol 3 (Oblivious Conversion):

- Input: $[\![a]\!]$ and (f_0, f_1, f_2, f_3).
- Output: f_a.

1. Using a copy protocol and rotation operations, generate $A = ([\![a]\!], [\![a-1]\!], [\![a-2]\!], [\![a-3]\!])$ from $[\![a]\!]$. Let W be the following sequence:

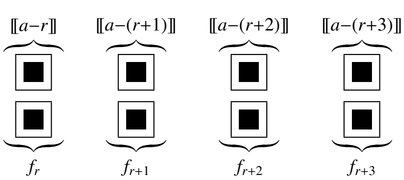


2. Apply a cyclic shuffle to W and obtain the following sequence:

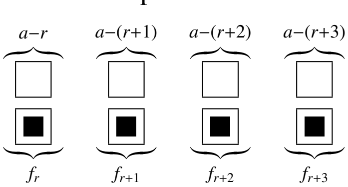


where r is the randomness used in the shuffle.

3. Decompose the stack as shown below:



4. Open the cards in the top line:



5. Output the stack under the card 0.

Theorem 3: *The above oblivious conversion protocol using m-sided polygon cards is secure. (It takes as inputs* $[\![a]\!]$ *and* $f_0, f_1, \cdots, f_{m-1}$*, and outputs* f_a*.) It uses two shuffles and* $m(k+1)+1$ *cards, where k is the number of cards contained in the stack* f_i*.*

Proof. Let A be a random variable of the input, and let V be a random variable of the visible sequence. Let E be a random variable of the opened value in the copy protocol of Step 1. Let R be a random variable of the randomness used in the cyclic shuffle used in Step 2. Let $E' = A - R \bmod 4$. As mentioned in the proof of Theorem 1, A and E are independent. Similarly, A and E' are independent. (The only difference is that the latter uses a cyclic shuffle but it does not affect this claim.) Moreover, A and (E, E') are also independent since E and E' are derived from independent and different shuffles. Thus, A and V are independent since V essentially consists of E, E'. Therefore, it is secure. □

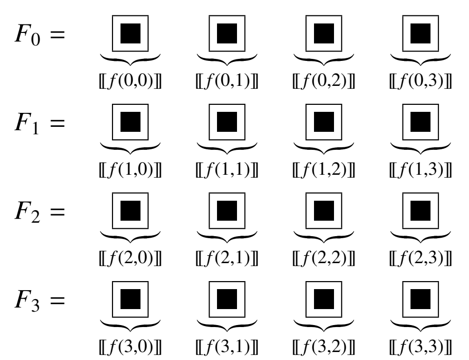
4.2 General Protocol

Using our oblivious conversion, Alice and Bob can securely compute an arbitrary function $f(x_1, x_2)$ whose input-domain and output-range are $\mathbb{Z}_m$.

Protocol 4 (Two-Party Protocol):

- Input: Alice has $a \in \mathbb{Z}_4$ and Bob has $b \in \mathbb{Z}_4$.
- Output: $[\![f(a, b)]\!]$.

1. Alice and Bob generate $[\![a]\!]$ and $[\![b]\!]$, respectively.
2. Alice and Bob place the following sequences F_0, F_1, F_2, F_3:



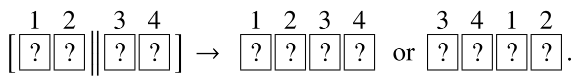
3. Let F'_i be a stack that is stacking of F_i. Using an oblivious conversion with inputs $[\![a]\!]$ and (F'_0, F'_1, F'_2, F'_3), they compute F'_a.
4. Let F_a be a sequence that is decomposing of F'_a. Using an oblivious conversion with inputs $[\![b]\!]$ and F_a, they compute $[\![f(a, b)]\!]$. This is the output of this protocol.

Theorem 4: *Let* $f : (\mathbb{Z}_m)^n \to \mathbb{Z}_m$ *be an arbitrary n-ary function. There is a secure protocol that takes as inputs* $([\![a_1]\!], \cdots, [\![a_n]\!])$ *and* $[\![f(x_1, \cdots, x_n)]\!]$ *for all* $x_1, \cdots, x_n \in \mathbb{Z}_m$*, and outputs* $[\![f(a_1, \cdots, a_n)]\!]$*. It uses* $2n$ *shuffles and* $m + n + m^n$ *cards.*

Proof. Extending the above protocol in a canonical way, it is possible to construct an n-party protocol. We first show that the protocol uses $m + n + m^n$ cards. The number of input cards is $n + m^n$. To copy $[\![a_1]\!]$, we needs m additional cards. On the other hand, we does not need additional cards to copy $[\![a_2]\!], \cdots, [\![a_n]\!]$ since the opened cards can be reused. Thus, the number of cards is $m + n + m^n$. Next we show the security of the protocol. Let A be a random variable of the input, and let V be a random variable of the visible sequence. For the i-th ($i = 1, 2, \cdots, n$) oblivious conversion, let E_{2i-1} be a random variable of the opened value in the copy protocol, and let E_{2i} be a random variable of the opened value in the last step. As mentioned in the proof of Theorem 3, A and (E_{2i-1}, E_{2i}) are independent. Since each random variable is independently derived from each shuffle, A and $(E_1, E_2, \cdots, E_{2n})$ are also independent. Thus, A and V are independent. Therefore, it is secure. □

4.3 Oblivious Conversion Using the Standard Cards

The oblivious conversion can also be applied to the standard cards (♣, ♡). We use the following standard encoding ♣♡ = 0 and ♡♣ = 1, and denote the face down encoding of a by $\mathsf{Com}(a)$. We also use a *random bisection cut* (which is denoted by $[\cdot\|\cdot]$) as below:

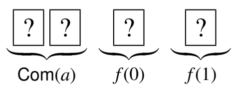


We note that it is derived from the cyclic shuffle by making stacks $1 \cdot 2$ and $3 \cdot 4$.

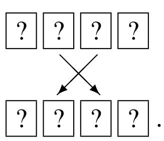
Protocol 5 (Oblivious Conversion Using Standard Cards):

- Input: $\mathsf{Com}(a)$ and two cards (or stacks) $f(0)$ and $f(1)$.
- Output: The card (or stack) $f(a)$.

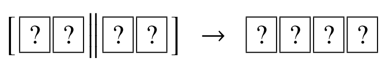
1. Place the cards as below.



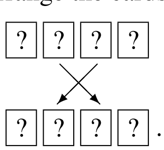
2. Rearrange the cards as below.



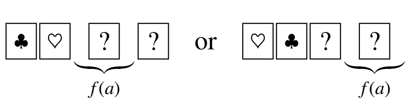
3. Apply a random bisection cut.



4. Rearrange the cards as below.



5. Open the first and second cards, then the output card $f(a)$ is obtained as follows.



Theorem 5: *The above oblivious conversion is secure. It uses one shuffle and* $2k + 2$ *cards, where* k *is the number of cards contained in* $f(0)$.

Proof. The opened value is independent of the inputs since the randomness used in the shuffle is chosen uniformly at random and independent of the inputs. Thus, it is secure. □

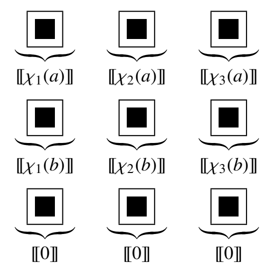
5. Voting Protocol for Multiple Candidates

In this section, we construct a voting protocol. Assume that there are n voters $A_1, \cdots, A_n$ and ℓ candidates $C_1, \cdots, C_\ell$. Each voter A_i has an input $a_i \in \{1, \cdots, \ell\}$. They wish to securely compute $c_i = \sum_{j=1}^{n} \chi_i(a_j)$, where $\chi_i(x) = 1$ if $x = i$, otherwise $\chi_i(x) = 0$.

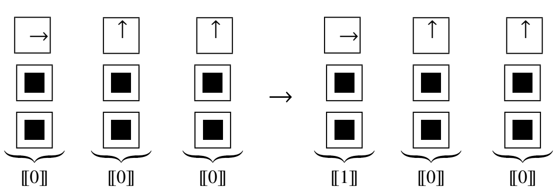
We will explicitly describe a voting protocol with two voters A, B and three candidates. The protocol takes as inputs A's input $a \in \{1, 2, 3\}$ and B's input $b \in \{1, 2, 3\}$, and outputs $(\llbracket\chi_1(a) + \chi_1(b)\rrbracket, \llbracket\chi_2(a) + \chi_2(b)\rrbracket, \llbracket\chi_3(a) + \chi_3(b)\rrbracket)$.

In the following, we will consider a simplified voting protocol which illustrates the idea behind and the correctness of the full protocol (Protocol 6). However, the simplified protocol does not hide which candidate each of the voters A and B vote for, and is hence not secure.

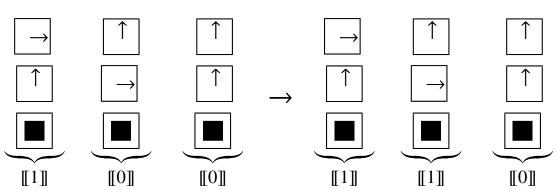
1. Place the cards as below:



2. Open the first row. Then, add one to the bottom-most card whose top card was $\llbracket 1\rrbracket$. For example, if the opening of the top row is as shown, then add one to the bottom-most card of the leftmost column:



3. Open the second row. Then, add one to the bottom-most card whose top card was $\llbracket 1\rrbracket$. For example, if the opening of the top row is as shown, then add one to the bottom-most card of the center column:



4. Output the bottom row.

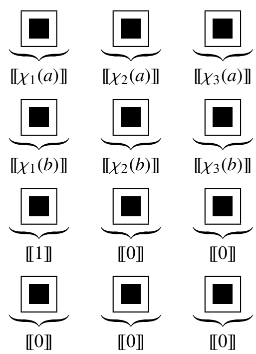
From the above description, it should be clear the simplified protocol correctly computes the voting result. However, as highlighted above, the protocol reveal which candidate each voter voted for.

In order to obtain the security, we use a cyclic shuffle. More concretely, we apply a cyclic shuffle to the sequence (d_1, d_2, d_3), where d_i is a stacking of the i-th column, and open the top row. Now the input is completely hidden due to the randomness of the cyclic shuffle. To keep track of the order of candidates when applying the cyclic shuffles, we append a sequence (⟦1⟧, ⟦0⟧, ⟦0⟧) which will be opened at the end of the protocol. The protocol proceeds as follows.

Protocol 6 (Voting Protocol):

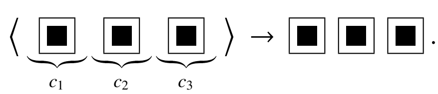
- Input: $a, b \in \{1, 2, 3\}$.
- Output: $(⟦y_1⟧, ⟦y_2⟧, ⟦y_3⟧)$ where $y_i = \chi_i(a) + \chi_i(b)$.

1. Place the cards as below:

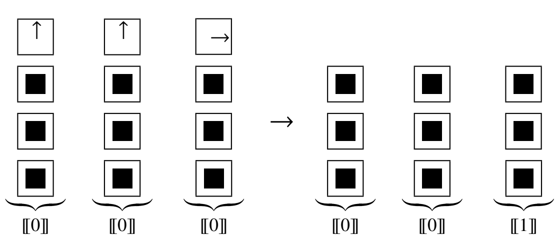


We use terms a *column* and a *row* in the usual sense. In this case, we have three columns and four rows.

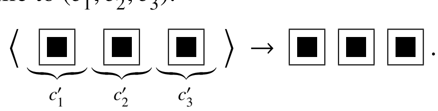
2. Make three stacks $c_1 = ⟦\chi_1(a)⟧ \cdot ⟦\chi_1(b)⟧ \cdot ⟦1⟧ \cdot ⟦0⟧$, $c_2 = ⟦\chi_2(a)⟧ \cdot ⟦\chi_2(b)⟧ \cdot ⟦0⟧ \cdot ⟦0⟧$ and $c_3 = ⟦\chi_3(a)⟧ \cdot ⟦\chi_3(b)⟧ \cdot ⟦0⟧ \cdot ⟦0⟧$. Apply a cyclic shuffle:



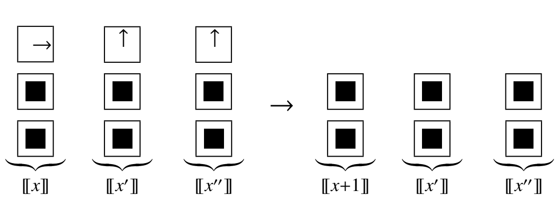
3. Open the top row and remove the top row. Then, add one to the bottom-most card whose top card was ⟦1⟧. For example, if the opening of the top row is as shown, then add one to the bottom-most card of the rightmost column:



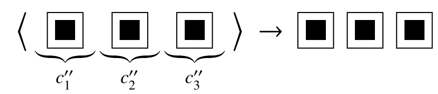
4. Let c'_1, c'_2 and c'_3 be the current columns. Apply a cyclic shuffle to (c'_1, c'_2, c'_3):



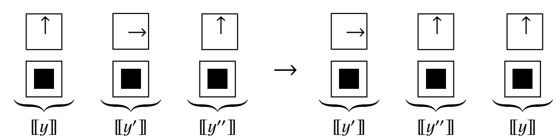
5. Open the top row and remove the top row. Then, add one to the bottom-most card whose top card was ⟦1⟧. For example, if the opening of the top row is as shown, then add one to the bottom-most card of the center column:



6. Let c''_1, c''_2 and c''_3 be the current columns. Apply a cyclic shuffle to (c''_1, c''_2, c''_3):



7. Open the top row. Rearrange the current sequence cyclically such that the column which has one in the top is the leftmost column. For example, if the opening of the top row is as shown, then rearrange as below:



8. Output the bottom row. The leftmost, center and rightmost cards correspond to the result values for the first, second and third candidates.

It is relatively straightforward to confirm that the changes done to the simplified protocol to obtain Protocol 6 will not change the output i.e. Protocol 6 will correctly compute the voting result. The following theorem will establish the security of Protocol 6.

Theorem 6: *Let $n, \ell \geq 1$. For n voters and ℓ candidates, the above voting protocol is secure. It uses $n+1$ shuffles and $(n+2)\ell$ cards.*

Proof. The opened values (in the above case, step 3, 5, and 7) are independent of the inputs since the randomnesses used in the shuffles are chosen uniformly at random and independent of the inputs. Thus, it is secure. □

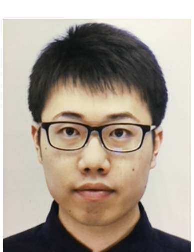
Acknowledgment

The authors would like to thank members of the study group "Shin-Akarui-Angou-Benkyou-Kai" for the valuable discussions and helpful comments. We also thank the editor and the anonymous reviewers, whose comments helped us to improve the presentation of this paper. This work was partially supported by JSPS KAKENHI Grant Numbers 26330001 and 26330151.

References

[1] C. Crépeau and J. Kilian, "Discreet solitary games," Advances in Cryptology - CRYPTO'93, vol.773 of Lecture Notes in Computer Science, pp.319–330, Springer, 1994.
[2] B. den Boer, "More efficient match-making and satisfiability: The five card trick," Advances in Cryptology - EUROCRYPT'89, vol.434 of Lecture Notes in Computer Science, pp.208–217, Springer, 1990.
[3] A. Koch, S. Walzer, and K. Härtel, "Card-based cryptographic protocols using a minimal number of cards," Advances in Cryptology - ASIACRYPT 2015, vol.9452 of Lecture Notes in Computer Science, pp.783–807, Springer, 2015.
[4] T. Mizuki, I.K. Asiedu, and H. Sone, "Voting with a logarithmic number of cards," Unconventional Computation and Natural Computation 2013, vol.7956 of Lecture Notes in Computer Science,

pp.162–173, Springer, 2013.
[5] T. Mizuki, U. Fumishige, and H. Sone, "Securely computing XOR with 10 cards," Australasian Journal of Combinatorics, 2006.
[6] T. Mizuki, M. Kumamoto, and H. Sone, "The five-card trick can be done with four cards," Advances in Cryptology - ASIACRYPT, vol.7658 of Lecture Notes in Computer Science, pp.598–606, Springer, 2012.
[7] T. Mizuki and H. Shizuya, "A formalization of card-based cryptographic protocols via abstract machine," Int. J. Inf. Sec., vol.13, no.1, pp.15–23, 2014.
[8] T. Mizuki and H. Shizuya, "Practical card-based cryptography," FUN 2014 Seventh International Conference on FUN WITH ALGORITHMS, vol.8496 of Lecture Notes in Computer Science, pp.313–324, Springer, 2014.
[9] T. Mizuki and H. Sone, "Six-card secure AND and four-card secure XOR," Third International Workshop on Frontiers in Algorithmics, vol.5598 of Lecture Notes in Computer Science, pp.358–369, Springer, 2009.
[10] V. Niemi and A. Renvall, "Secure multiparty computations without computers," Theor. Comput. Sci., vol.191, no.1-2, pp.173–183, 1998.
[11] T. Nishida, Y. Hayashi, T. Mizuki, and H. Sone, "Card-based protocols for any Boolean function," Theory and Applications of Models of Computation, TAMC 2015, vol.9076 of Lecture Notes in Computer Science, pp.110–121, Springer, 2015.
[12] T. Nishida, T. Mizuki, and H. Sone, "Securely computing the three-input majority function with eight cards," 2nd International Conference on the Theory and Practice of Natural Computing, TPNC 2013, vol.8273 of Lecture Notes in Computer Science, pp.193–204, Springer, 2013.
[13] A. Nishimura, Y. Hayashi, T. Mizuki, and H. Sone, "An implementation of non-uniform shuffle for secure multi-party computation," Proc. 3rd ACM International Workshop on ASIA Public-Key Cryptography, AsiaPKC@AsiaCCS, pp.49–55, 2016.
[14] K. Shinagawa, "Oblivious conversion using 4-sided cards," YouTube, 2015. https://youtu.be/hlAetm66iRU
[15] K. Shinagawa, "Secure addition protocol using 4-sided cards," YouTube, 2015. https://youtu.be/9Tid6X-9r-c
[16] K. Shinagawa, T. Mizuki, J. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto, "Multi-party computation with small shuffle complexity using regular polygon cards," The 9th International Conference on Provable Security, ProvSec, pp.127–146, 2015.
[17] K. Shinagawa, T. Mizuki, J. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto, "Secure multi-party computation using polarizing cards," The 10th International Workshop on Security, IWSEC, pp.281–297, 2015.
[18] A. Stiglic, "Computations with a deck of cards," Theor. Comput. Sci., vol.259, no.1-2, pp.671–678, 2001.

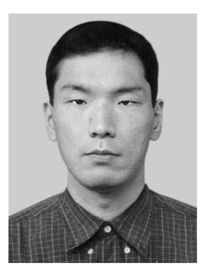


Kazumasa Shinagawa received his B.E. degree from University of Tsukuba in 2015. He is a master course student of University of Tsukuba. He received SCIS Best Paper Award from IEICE in 2015 and CSS Best Student Paper Award from IPSJ in 2015.

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 Anniversary 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.