International Journal of Information Security (2022) 21:833–846
https://doi.org/10.1007/s10207-022-00585-8

REGULAR CONTRIBUTION

Coin-based Secure Computations

Yuichi Komano[1] · **Takaaki Mizuki**[2]

Published online: 6 April 2022

Abstract
In the history of cryptography, many cryptographic protocols have relied on random coin tosses to prove their security. Although flipping coins is indispensable in this manner, the coins themselves have never been in the spotlight. Therefore, we would like to make *physical coins* rise to the level of cryptography, just as a *deck of physical playing cards* has been used to perform a secure multi-party computation. Such a card-based protocol is known to be helpful both to perform a secure computation without any black-box computers and to understand the principles of secure protocols. In this paper, we propose a new framework of secure multi-party computation using physical coins, named a *coin-based protocol*. One advantage of the use of coins is that they are more ubiquitous than cards. Whereas a face-down card can conceal the information about its face side, one side of a coin reveals the information of its other side. Hence, more careful design is required for a secure coin-based protocol than for a card-based one. We formalize a computational model of the coin-based protocol and explicitly give protocols for NOT, AND, COPY, OR, and XOR computations. We also discuss the composability of the extended protocols and how to implement them in practice.

Keywords Multi-party computation · Card-based protocol · Physical coin

1 Introduction

Physical coins are widely used to perform random and fair selections, such as coin tosses in sports games. In the research of cryptography, coins have conceptually appeared as "*the probability is taken over the random coin toss*." Although coins play an important but invisible role, the coins themselves have never been on the center stage of cryptographic research. Therefore, we would like to make coins rise to this level. In contrast, a *deck of physical playing cards* has received the spotlight. That is, designing card-based protocols, which perform secure multi-party computations [6] using physical cards, is an active research area. Let us start with a review of card-based protocols.

1.1 Related Work: Card-based Protocol

The first card-based protocol was proposed by den Boer [3] in 1989. His protocol performs a secure AND computation with five physical cards. Since the five-card AND protocol was invented, many protocols using fewer cards or realize other useful functions have been developed [4,7,14,16,19,20,22]. In addition, a computational model [17] and its implementation [23] have been established. Refer to [18] for a survey of recent the recent progress on card-based protocols.

In a card-based protocol, a two-suit deck of cards, for example, $\boxed{\clubsuit}$, $\boxed{\heartsuit}$ are typically used. For example, in the six-card AND protocol [19], each player first places face-down cards $\boxed{?}\boxed{?}$ in accordance with his or her private input, based on the encoding $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. Such a pair of face-down cards is called a commitment and the protocol starts with two commitments $\boxed{?}\boxed{?}\boxed{?}\boxed{?}$ placed by two players, together with two more cards $\boxed{\clubsuit}\boxed{\heartsuit}$. (For example, the initial sequence is $\Gamma^{01} = \left(\boxed{\tfrac{?}{\clubsuit}}, \boxed{\tfrac{?}{\heartsuit}}, \boxed{\tfrac{\clubsuit}{?}}, \boxed{\tfrac{\heartsuit}{?}}, \boxed{\tfrac{?}{\heartsuit}}, \boxed{\tfrac{?}{\clubsuit}}\right)$ for $a = 0$ and $b = 1$, where $\boxed{\tfrac{?}{\clubsuit}}$ and $\boxed{\tfrac{?}{\heartsuit}}$ denote face-down cards,

An earlier version of this study was presented at the 7th International Conference on the Theory and Practice of Natural Computing, TPNC 2018, Dublin, Ireland, December 12–14, 2018, and appeared in Proc. TPNC 2018, Lecture Notes in Computer Science, Springer International Publishing, vol. 11324, pp. 87–98, 2018 [10].

✉ Yuichi Komano
yuichi1.komano@toshiba.co.jp

Takaaki Mizuki
tm-paper+coin@g-mail.tohoku-university.jp

[1] Toshiba Corporation, 1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, Japan

[2] Tohoku University, 6-3 Aramaki-Aza-Aoba, Aoba-ku, Sendai, Japan

Fig. 1 Examples of coins (left) and creating a coin-based commitment (right two)

and $\frac{\clubsuit}{?}$ and $\frac{\heartsuit}{?}$ denote face-up cards.) Then, they change the order of the cards by a series of card operations, such as *rearrangements* and *shuffles*, to obtain a commitment to the AND value. Because these operations are easy to implement and both their correctness and security are intuitively understandable, such card-based protocols have been widely used to solve social problems in daily life, as well as to educate non-experts about cryptography [12,18].

One drawback of card-based protocols is that people do not typically carry a deck of cards to solve social problems. In contrast, many people have physical coins in their pockets or wallets. Thus, it would be beneficial to make use of such coins for secure multi-party computations.

1.2 Our Contribution

In this paper, we introduce a framework for multi-party computation that uses physical coins. We also give concrete examples of the coin-based protocols, such as an AND protocol. We assume that the head and tail of each coin have different patterns, as depicted in Fig. 1. Throughout this paper, we assume that the design of every coin is the same and that nobody can distinguish one coin from another[1].

Hereinafter, ○ and ● denote a face-up coin (head) and a face-down coin (tail), respectively. For example, the first (leftmost) and second coins in Fig. 1 are denoted by ○ and ●, respectively. Given a single coin, we can encode a one-bit value with ○ and ● as

$$\bullet = 0 \textit{ and } \circ = 1. \tag{1}$$

Recall that in a *card-based* protocol, two players, say Alice and Bob, place face-down cards according to their private input values; these values can be kept secret because the backside $\boxed{?}$ of every card has no information. In contrast, a coin placed on a table reveals, from either side, the information of the other side. This means that, if we simply replace a card with a coin in a card-based protocol, say, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ with ● and ○, respectively, then the resulting "coin-based" protocol is no longer secure. To construct a secure coin-based protocol, new ideas for implementation and a computational model are required. Our answer is to hide the surface of the coin by having the player grab it and hold it in their hand or by stacking another coin onto the coin. For example, as illustrated in Fig. 1, a player can create a "coin-based" commitment to her or his private input bit by grabbing a coin without anyone else seeing which side is up. We give a formal treatment for coin-based protocols and their security in this paper.

In addition, we show concrete examples of coin-based protocols for NOT, AND, COPY, OR, and XOR computations, consisting of action sequences. We then confirm their correctness and security with the probability trace and the extended diagram proposed in [15]. We also discuss the implementation of actions and show that our protocols are executable in practice.

We emphasize that there are three merits to our coin-based protocols:

1 As we review in Section 1.1, there are many studies on card-based protocols. In addition to card-based protocols, other protocols use a physical tool, such as, protocols with a PEZ dispenser [1,2], tamper-evident seals [21], and visual secret sharing sheets [5], and bags and balls [13]. Compared with the physical objects in these protocols, a coin is a simple tool representing just one-bit information by itself. Specifically, with such a simple tool, the coin-based protocol can provide the most fundamental protocol with familiar tools. Its computation model or procedure can be useful in developing protocols with other tools.
2 Similar to the card-based protocol, it is easy to trace the steps of the coin-based protocol by tracking the actions of players' hands. Moreover, intuitively, it is also easy to check whether there is information leakage. Hence, the coin-based protocol is useful for understanding the principles of information security and cryptology. That is, the coin-based protocol, with a coin as a simpler and more familiar tool compared with other protocols, is a good tool for educating not only students but also new researchers and engineers of information security, and for enlightening citizens.
3 As with the card-based protocol, the coin-based protocol is executed without *any black-box computers or any network communication*. Hence, as players perform the coin-based protocols, they are reassured that there is no unintended leakage of their secrets, such as spyware surreptitiously transferring hidden information.

The main difference from the conference version of this paper [10] is that we add, as new material, Sections 3.4, 3.5, 4, and 5. In Sections 3.4 and 3.5, we give two basic coin-based protocols, namely, OR and XOR protocols, respectively. In

[1] If coins have additional information, such as the year of manufacture in metallic currencies, we should use coins with the same exact design, such as coins made in the same year.

Section 4.1, we formalize another action, pick, which is necessary for the protocol composition. We also modify the inputs of our coin-based protocols to be suitable for protocol composition, and give examples of protocol compositions, namely, a three-input AND protocol and a three-input majority decision protocol in Sections 4.2 and 4.3, respectively. Furthermore, in Section 5, we add discussions on the effect of human errors, namely, the correctness of the protocol and the information leakage of players' private inputs.

1.3 Organization

The remainder of this paper is organized as follows. In Section 2, we present our idea behind formalizing coin-based protocols and their definitions, and show that our actions are implementable. We then show concrete examples of coin-based protocols in Section 3. In Section 4, we introduce another action, pick, and give examples of protocol compositions. Moreover, following the discussion of human error in the card-based protocol [15], we discuss the effect of human error on the coin-based protocol in Section 5. Finally, Section 6 concludes this paper.

2 Model of Coin-based Protocols

In this section, we first present the idea behind our construction of coin-based protocols and then define their computational model formally. We also discuss how to implement the actions appearing in coin-based protocols.

2.1 Basic Idea

As mentioned in Section 1.2, given an existing card-based protocol, we can immediately transform it into an (insecure) coin-based protocol, by replacing the cards with coins, that is, by replacing $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ with $\bullet$ and $\circ$, respectively. For example, if we execute the six-card AND protocol mentioned in Section 1.1 with a sequence of coins, such as $\Gamma^{01} = (\bullet, \circ, \bullet, \circ, \circ, \bullet)$ for $a = 0$ and $b = 1$, we obtain a pair of coins corresponding to the value of $a \wedge b$.

The above coin-based protocol is insecure because a single coin cannot hold any secret information. How can we make coins behave more like cards? If we place a dummy coin on a given coin, the stack of both coins can simulate a card. Therefore, if we use twice as many coins as the number of cards used in a card-based protocol, we can construct a secure coin-based protocol. For example, let us put $\circ$ on all six coins in $\Gamma^{01} = (\bullet, \circ, \bullet, \circ, \circ, \bullet)$. Because only the dummy coins are visible during the execution of the protocol, no secret information is revealed.

Technically, such a coin-based protocol using dummy coins works correctly and securely, but it may be hard for humans to use a stack of coins as if they were a card. Therefore, we should make use of more human-friendly actions. For instance, as already seen in Fig. 1, a player can create a commitment by grabbing a coin, which is an easy action for humans. Thus, our questions are:

1 Can we construct human-friendly coin-based protocols?
2 How can we model such a coin-based protocol formally?

As an answer, we formalize the computational model with five actions in the next subsection.

2.2 Coin-based Protocols

As stated in Section 1.2, let $\circ$ and $\bullet$ denote face-up and face-down coins, respectively. As also stated, we assume that all coins are indistinguishable from each other by their surface appearance.

For two coins $c, c' \in \{\circ, \bullet\}$, let a stack of coins, where c is on c', be denoted by cc'. For example, $cc' = \circ\bullet$ is a stack of two coins for $c = \circ$ and $c' = \bullet$, such that the top coin of the stack is head up and the bottom coin is tail up. For more than two coins, we consider a stack in a similar manner. For two stacks of two coins $\mathbf{c_1} = \circ\bullet$ and $\mathbf{c}_2 = \bullet\circ$, for instance, $\mathbf{c}_1\mathbf{c}_2$ is the stack of four coins $\circ\bullet\bullet\circ$.

$\mathcal{S}^k$ denotes the set of all stacks of at most k coins for some integer k, namely, let

$$\begin{aligned}\mathcal{S}^k &= \{\epsilon\} \cup \bigcup_{i=1}^{k} \{\circ, \bullet\}^i \\ &= \{\epsilon, \circ, \bullet, \circ\circ, \circ\bullet, \bullet\circ, \bullet\bullet, \circ\circ\circ, \circ\circ\bullet, \cdots\},\end{aligned}$$

which is a finite set of all strings over the alphabets $\{\circ, \bullet\}$. Here, we designate the symbol ϵ as an empty stack with no coin.

Let us consider a coin-based protocol to be executed by two *semi-honest* players, Alice and Bob, with a table and k coins. We use a tuple to describe the status of the coins, which we call an *arrangement*, during execution of the protocol:

$$(\mathbf{a}_L, \mathbf{a}_R | \mathbf{b}_L, \mathbf{b}_R | \mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_k),$$

where, $\mathbf{a}_L, \mathbf{a}_R, \mathbf{b}_L, \mathbf{b}_R, \mathbf{t}_i \in \mathcal{S}^k$ are stacks of coins in (closed) Alice's left hand, Alice's right hand, Bob's left hand, Bob's right hand, and the i-th area of the table where $1 \leq i \leq k$, respectively. We assume that every hand is closed. Then, $\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R$, and $\mathbf{T}_i$ denote their variables, namely, the locations where stacks of coins are placed. We use this notation to define the set of all arrangements of stacks from k coins as

$$\mathsf{Arg}^k = \Big\{ (\mathbf{c}_1, \mathbf{c}_2|\mathbf{c}_3, \mathbf{c}_4|\mathbf{c}_5, \mathbf{c}_6, \cdots, \mathbf{c}_{k+4}) \in \left(\mathcal{S}^k\right)^{k+4} : \sum_{i=1}^{k+4} \mathsf{size}(\mathbf{c}_i) = k \Big\},$$

where $\mathsf{size}(\mathbf{c}_i)$ means the size of $\mathbf{c}_i$, namely, the number of coins in $\mathbf{c}_i$, which is defined by

$$\mathsf{size}(c_1 c_2 \cdots c_n) = n \text{ and } \mathsf{size}(\epsilon) = 0$$

for a stack of n coins $c_1 c_2 \cdots c_n \in \mathcal{S}^k$ ($c_i \in \{\circ, \bullet\}$).

As seen below, we use $U \subseteq \mathsf{Arg}^k$ to represent a set of initial arrangements, that is, inputs of the protocol. In the subsequent operations, we may omit $\mathbf{t}_j$ in an arrangement and the corresponding variable $\mathbf{T}_j$ if a protocol requires k' areas on the table, and no coin is placed on the j-th area ($j > k'$) throughout the protocol.

For a stack of coins $\mathbf{c} \in \mathcal{S}^k$, $\mathsf{top}(\mathbf{c})$, $\mathsf{bottom}(\mathbf{c})$, and $\mathsf{turn}(\mathbf{c})$ denote the top of $\mathbf{c}$, the bottom of $\mathbf{c}$, and the turned stack of $\mathbf{c}$, respectively (they are defined to be ϵ if $\mathbf{c} = \epsilon$). Namely, for a stack of n coins $\mathbf{c} = c_1 c_2 \cdots c_n \in \mathcal{S}^k$ ($c_i \in \{\circ, \bullet\}$),

$$\mathsf{top}(\circ c_2 \cdots c_n) = \circ, \mathsf{top}(\bullet c_2 \cdots c_n) = \bullet,$$
$$\mathsf{bottom}(c_1 c_2 \cdots c_{n-1} \circ) = \bullet, \mathsf{bottom}(c_1 c_2 \cdots c_{n-1} \bullet) = \circ,$$
$$\mathsf{turn}(c_1 c_2 \cdots c_n) = \mathsf{turn}(c_n) \cdots \mathsf{turn}(c_2)\mathsf{turn}(c_1), \text{ and}$$
$$\mathsf{top}(\epsilon) = \mathsf{bottom}(\epsilon) = \mathsf{turn}(\epsilon) = \epsilon,$$

where $\mathsf{turn}(\circ) = \bullet$ and $\mathsf{turn}(\bullet) = \circ$.

Let us define a visible sequence for an arrangement $\Gamma = (\mathbf{c}_1, \mathbf{c}_2|\mathbf{c}_3, \mathbf{c}_4|\mathbf{c}_5, \mathbf{c}_6, \cdots, \mathbf{c}_{k+4}) \in \mathsf{Arg}^k$. We first extend top as follows. For $\mathbf{c}_i \in \mathcal{S}^k$, $1 \le i \le 4$, in such Γ, $\overline{\mathsf{top}}(\mathbf{c}_i)$ is "?" if a stack of coins is in Alice's or Bob's (closed) hand, that is, $\mathbf{c}_i \neq \epsilon$; otherwise, it is ϵ. Namely, for $\mathbf{c}_i \in \mathcal{S}^k$,

$$\overline{\mathsf{top}}(\mathbf{c}_i) = \begin{cases} ? & (i \in [1,4] \text{ and } \mathbf{c}_i \neq \epsilon) \\ \epsilon & (i \in [1,4] \text{ and } \mathbf{c}_i = \epsilon) \\ \mathsf{top}(\mathbf{c}_i) & (i \ge 5) \end{cases},$$

where $[i, j]$ denotes a set of integers $\{x \in \mathbb{Z} | i \le x \le j\}$ for integers i and j.

Now, for the above Γ, we set

$$\overline{\mathsf{top}}(\Gamma) = (\overline{\mathsf{top}}(\mathbf{c}_1), \overline{\mathsf{top}}(\mathbf{c}_2)|\overline{\mathsf{top}}(\mathbf{c}_3), \overline{\mathsf{top}}(\mathbf{c}_4)| \overline{\mathsf{top}}(\mathbf{c}_5), \overline{\mathsf{top}}(\mathbf{c}_6), \cdots, \overline{\mathsf{top}}(\mathbf{c}_{k+4})),$$

which we call the *visible sequence* of Γ. For example, if $\mathbf{c}_1 = \mathbf{c}_5 = \circ\bullet$, $\mathbf{c}_2 = \mathbf{c}_4 = \mathbf{c}_6 = \bullet\circ$ and $\mathbf{c}_3 = \epsilon$, that is, $\Gamma = (\circ\bullet, \bullet\circ|\epsilon, \bullet\circ|\circ\bullet, \bullet\circ)$, $\overline{\mathsf{top}}(\Gamma) = (?, ?|\epsilon, ?|\circ, \bullet)$. Furthermore, the *set of all visible sequences* (of Γ with k coins) is defined as

$$\mathsf{Vis}^k = \left\{ \overline{\mathsf{top}}(\Gamma) : \Gamma \in \mathsf{Arg}^k \right\}.$$

With these notations, let us formally define a coin-based protocol.

Definition 1 (Coin-based protocol) A *coin-based protocol* is specified with a quadruple $\mathcal{P} = (k, U, Q, A)$:

- k is the number of coins used in the protocol;
- U is an *input set* where $U \subseteq \mathsf{Arg}^k$;
- Q is a *state set* having an *initial state* $q_0 \in Q$ and a *final state* $q_f \in Q$;
- $A : (Q - \{q_f\}) \times \mathsf{Vis}^k \to Q \times \mathsf{Action}$ is an *action function*, where Action is the set of the following actions:
  - $(\mathsf{move}, \mathbf{P}_1 \to \mathbf{P}_2, n)$ for $\mathbf{P}_1, \mathbf{P}_2 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R, \mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$ with $\mathbf{P}_1 \neq \mathbf{P}_2$: A player moves the upper n coins of the stack $\mathbf{p}_1$, consisting of $m (\ge n)$ coins, on $\mathbf{P}_1$ onto $\mathbf{p}_2$ on $\mathbf{P}_2$. Let $\mathbf{p}^{(u)}$ and $\mathbf{p}^{(l)}$ denote the stack of the upper n coins and lower $m - n$ coins of $\mathbf{p}$, respectively. This action changes the arrangement from $(\cdots, \mathbf{p}_1, \cdots, \mathbf{p}_2, \cdots)$ to $(\cdots, \mathbf{p}_1^{(l)}, \cdots, \mathbf{p}_1^{(u)}\mathbf{p}_2, \cdots)$.
    If $\mathbf{P}_1 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\}$, the player opens her or his hand at first. If another stack $\mathbf{p}_2$ is in $\mathbf{P}_2 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\}$, she or he opens their hand and moves $\mathbf{p}_1^{(u)}$ onto $\mathbf{p}_2$. At the end, she or he closes their hands. Note that, when Alice and Bob hold stacks of coins in all their hands in $\{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\} \backslash \{\mathbf{P}_1, \mathbf{P}_2\}$, it is infeasible to execute this action; hence, the protocol stops (fails) in this case. Also note that $\mathsf{top}(\mathbf{p}_1)$, $\mathsf{top}(\mathbf{p}_1^{(l)})$, and $\mathsf{top}(\mathbf{p}_2)$ are visible to the public. The players perform operates this action so that no information leaks except the visible coins.
  - $(\mathsf{hand}, \mathbf{P}_2 \leftarrow \mathbf{P}_1)$ for $\mathbf{P}_1, \mathbf{P}_2 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\}$ with $\mathbf{P}_1 \neq \mathbf{P}_2$: A player puts a hand $\mathbf{P}_1$ holding a stack of coins $\mathbf{p}_1 \in \mathcal{S}^k$ on another hand $\mathbf{P}_2$ so that the palms of both hands touch each other. Then, the players open their hands at the same time, and close their bottom hand so that the composite stack is hidden in the closed hand. This action changes the arrangement from $(\cdots, \mathbf{p}_1, \cdots, \mathbf{p}_2, \cdots)$ to $(\cdots, \epsilon, \cdots, \mathsf{turn}(\mathbf{p}_1)\mathbf{p}_2, \cdots)$.
  - $(\mathsf{shuffle}, \mathbf{P}_1, \mathbf{P}_2)$ for $\mathbf{P}_1, \mathbf{P}_2 \in \{\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$ with $\mathbf{P}_1 \neq \mathbf{P}_2$: A player shuffles the two stacks placed on $\mathbf{P}_1$ and $\mathbf{P}_2$. This action changes the arrangement from $(\cdots, \mathbf{p}_1, \cdots, \mathbf{p}_2, \cdots)$ to $(\cdots, \mathbf{p}_1, \cdots, \mathbf{p}_2, \cdots)$ (which is unchanged) or $(\cdots, \mathbf{p}_2, \cdots, \mathbf{p}_1, \cdots)$, where each case occurs with a probability of $\frac{1}{2}$. No player can know which case occurs if $\mathsf{size}(\mathbf{p}_1) = \mathsf{size}(\mathbf{p}_2)$ and $\mathsf{top}(\mathbf{p}_1) = \mathsf{top}(\mathbf{p}_2)$. Note that, unless at least one of Alice and Bob holds no stack of coins

Fig. 2 Example of implementation of hand

in both hands, it is infeasible to execute this action; hence, the protocol stops (fails) when both players hold a stack of coins in her or his hand.

- (flip, **P**) for $\mathbf{P} \in \{\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$: A player turns over the stack on **P**. This action changes the arrangement from $(\cdots, \mathbf{p}, \cdots)$ to $(\cdots, \mathsf{turn}(\mathbf{p}), \cdots)$. Note that, unless at least one of Alice and Bob holds no stack of coins in her or his hand, it is infeasible to execute this action; hence, the protocol stops (fails) when both players hold stacks of coins in their hands.
- (rflip, **P**) for $\mathbf{P} \in \{\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$: A player randomly flips the stack placed on **P**. This action changes the tuple from $(\cdots, \mathbf{p}, \cdots)$ to $(\cdots, \mathbf{p}, \cdots)$ (which is unchanged) or $(\cdots, \mathsf{turn}(\mathbf{p}), \cdots)$; each case occurs with a probability of $\frac{1}{2}$. No player can know which case occurs if $\mathsf{top}(\mathbf{p}) = \mathsf{bottom}(\mathbf{p})$. Note that, similar to shuffle, unless at least one of Alice and Bob holds no stack of coins in both hands, it is infeasible to execute this action; hence, the protocol stops (fails) when both players hold a stack of coins in her or his hand.

We say that the protocol for a function f is *correct* if it finally outputs the correct value of $f(a, b)$ for any input (a, b).

The protocol $\mathcal{P} = (k, U, Q, A)$ proceeds as a Turing machine does. That is, starting from an initial state q_0 and an initial arrangement $\Gamma_0 \in U$, its current state q and arrangement Γ move to the next state q' and arrangement Γ', respectively, according to the output of the action function A.

2.3 Feasibility of Actions

In this subsection, we discuss the implementation of the five actions in Definition 1. Among these five actions, move and flip are naturally implementable without explanation. Therefore, we focus on the remaining three actions; hand, shuffle, and rflip.

Figure 2 shows an example of the implementation of hand, consisting of five steps. Initially, each player holds stacks of coins in both hands (top left in Fig. 2). Then, they each place one hand on top of the other to move (and overturn) the stack in the upper hand onto the stack in the lower hand (top middle). After that, both players open both hands to pile up the stacks under the upper hand (top right). After the stacks are piled up, both players close their lower hand to hide the stack (lower left). Subsequently, both remove their upper hand (lower middle). Finally, they open the removed hand to show that there is no coin (lower right). Note that, after this action, the stack of coins in the upper hand becomes upside down. For example, the operation $(\mathsf{hand}, A_L \leftarrow B_L)$ changes the arrangement $(\circ, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon)$ to $(\mathsf{turn}(\circ\bullet)\circ, \circ|\epsilon, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon)$, which is equal to $(\circ \bullet \circ, \circ|\epsilon, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon)$ because $\mathsf{turn}(\circ\bullet) = \mathsf{turn}(\bullet)\mathsf{turn}(\circ) = \circ\bullet$.

As for shuffle, it can be operated in a similar manner to the shuffling operation in the card-based protocol. A player exchanges the positions of two stacks of coins multiple times so that the number of moves cannot be traced.

We then show two implementations for rflip. One of them is performed without any item and the other uses an item such as a binder clip. In the former implementation, a player holds the stack of coins and rotates it horizontally multiple times so that the number of the rotations cannot be traced, as shown in the left of Fig. 3. In the latter case, the player clips stacks into one stack with, for example, a binder clip, as shown in the right side of Fig. 3, and then, throws the clipped stack in the air like a coin toss.

Note that, to make the results shuffle and flip random from the viewpoint of both players, they can execute the action in relays (sequentially).

In addition, it is natural to assume that a semi-honest player is able to create a coin-based commitment (according to her or his private bit) by adjusting the direction of the coin after grabbing it.

Fig. 3 Examples of implementations of rflip

2.4 Definitions for Security

This subsection gives definitions related to the security of the protocols. As we explain in Section 2.2, we assume that the players are semi-honest. Let us assume that other entities, *including an adversary*, are also semi-honest. That is, the aim of an attacker is to maliciously obtain any information from the secret input, by observing the protocol.

To discuss security, we need to consider *publicly observable information*, which may leak a secret input with a protocol. There are two kinds of such information. The first one is visible information of coins on the table, such as the surface (direction) of the topmost coin and the number of stacked coins. The second one is publicly detectable information of coins in a player's hand, such as the coin of an initial arrangement, that is independent from the input, and one which is publicly detectable from the previous state of the protocol.

Let us define a *detectable sequence* for an arrangement $\Gamma = (\mathbf{c}_1, \mathbf{c}_2|\mathbf{c}_3, \mathbf{c}_4|\mathbf{c}_5, \mathbf{c}_6, \cdots, \mathbf{c}_{k+4}) \in \mathsf{Arg}^k$. We first extend top and size as follows. Unlike $\overline{\mathsf{top}}(\mathbf{c})$ which returns "?" if $\mathbf{c}$ is in a player's hand, the following $\widetilde{\mathsf{top}}(\mathbf{c})$ and $\widetilde{\mathsf{size}}(\mathbf{c})$ return invisible information if it is publicly detectable from the specification of the protocol. That is, for $\mathbf{c}_i \in \mathcal{S}^k$,

$$\widetilde{\mathsf{top}}(\mathbf{c}_i) = \begin{cases} \mathsf{top}(\mathbf{c}_i) & (i \in [1,4] \text{ and it is detectable}) \\ ? & (i \in [1,4] \text{ and it is not detectable yet}) \\ \mathsf{top}(\mathbf{c}_i) & (i \geq 5) \end{cases}, \text{ and}$$

$$\widetilde{\mathsf{size}}(\mathbf{c}_i) = \begin{cases} \mathsf{size}(\mathbf{c}_i) & (i \in [1,4] \text{ and it is detectable}) \\ ? & (i \in [1,4] \text{ and it is not detectable yet}) \\ \mathsf{size}(\mathbf{c}_i) & (i \geq 5) \end{cases}.$$

We then set the *detectable sequence* of Γ, $\left(\widetilde{\mathsf{top}}(\Gamma), \widetilde{\mathsf{size}}(\Gamma)\right)$, with

$$\widetilde{\mathsf{top}}(\Gamma) = \left(\widetilde{\mathsf{top}}(\mathbf{c}_1), \widetilde{\mathsf{top}}(\mathbf{c}_2)|\widetilde{\mathsf{top}}(\mathbf{c}_3), \widetilde{\mathsf{top}}(\mathbf{c}_4)|\widetilde{\mathsf{top}}(\mathbf{c}_5), \widetilde{\mathsf{top}}(\mathbf{c}_6), \cdots, \widetilde{\mathsf{top}}(\mathbf{c}_{k+4})\right) \text{ and}$$

$$\widetilde{\mathsf{size}}(\Gamma) = \left(\widetilde{\mathsf{size}}(\mathbf{c}_1), \widetilde{\mathsf{size}}(\mathbf{c}_2)|\widetilde{\mathsf{size}}(\mathbf{c}_3), \widetilde{\mathsf{size}}(\mathbf{c}_4)|\widetilde{\mathsf{size}}(\mathbf{c}_5), \widetilde{\mathsf{size}}(\mathbf{c}_6), \cdots, \widetilde{\mathsf{size}}(\mathbf{c}_{k+4})\right).$$

For example, assume that $\mathbf{c}_1 = \mathbf{c}_5 = \circ\bullet$, $\mathbf{c}_2 = \mathbf{c}_4 = \mathbf{c}_6 = \bullet\circ$, and $\mathbf{c}_3 = \epsilon$, that is, $\Gamma = (\circ\bullet, \bullet\circ|\epsilon, \bullet\circ|\circ\bullet, \bullet\circ)$. If coins in Alice's hands are undetectable (but the number of coins are detectable), and if ones in Bob's hands are detectable, $\widetilde{\mathsf{top}}(\Gamma) = (?, ?|\epsilon, \bullet|\circ, \bullet)$ and $\widetilde{\mathsf{size}}(\Gamma) = (2, 2|0, 2|2, 2)$.

To confirm the correctness and security of a coin-based protocol, we used an extended diagram [15] that replaces a probability within the diagram, as proposed by Koch, Walzer, and Härtel [7], with *the probability trace*. We give a definition of the probability trace below, which is a modification from [15] to replace the step number j to the *detectable sequence trace* d for Step j of the protocol. Here, the detectable sequence trace for Step j is a set of detectable sequences which appear before or at Step j of the protocol.

Definition 2 (Probability trace) Let $|U|$ be the number of elements in an input set $U = \left\{\Gamma_0^1, \Gamma_0^2, \cdots, \Gamma_0^{|U|}\right\}$ of a coin-based protocol $\mathcal{P}$. Let d be a detectable sequence trace. An $|U|$-tuple $(q_{1,d,s}, q_{2,d,s}, \cdots, q_{|U|,d,s})$ such that

$$q_{i,d,s} = \Pr\left[M = \Gamma_0^i, G_v = s | D = d\right]$$

is called a *probability trace* for an arrangement s and the detectable sequence trace d, where M, G_j, and D are random variables of the original input sequence of the arrangement when d is seen, and of the detectable sequence trace, respectively.

With the detectable sequence trace and probability trace, let us define the security of the coin-based protocol as follows.

Definition 3 (Perfect security of coin-based protocol) We say that a coin-based protocol $\mathcal{P}$ is *perfectly secure* if it leaks no information for any run of the protocol (i.e., the input and the detectable sequence trace are independent).

Note that an implementation of a coin-based protocol may leak the number of held coins ($\mathbf{c}$ where $\widetilde{\mathsf{size}}(\mathbf{c}) = ?$) via *side-channel information*[2], such as the volume of players' hands and the jingle of coins in players' hands. Depending on the protocol, the number of held coins may be different based on the secret input, and, in this case, such side-channel information may leak a secret, that is, the protocol may be insecure. In our protocols presented later, the number of held coins does not depend on the secret input and no secret leaks from the side-channel information. For the sake of simplicity, we hereafter ignore such information leakage in this paper.

[2] In the implementation of a cryptographic protocol, it is known that *side-channel information*, such as the timing information and the power consumption, can leak a secret from the protocol, *a.k.a., side-channel attacks* [8,9]. Against such attacks, many countermeasures have been discussed [11].

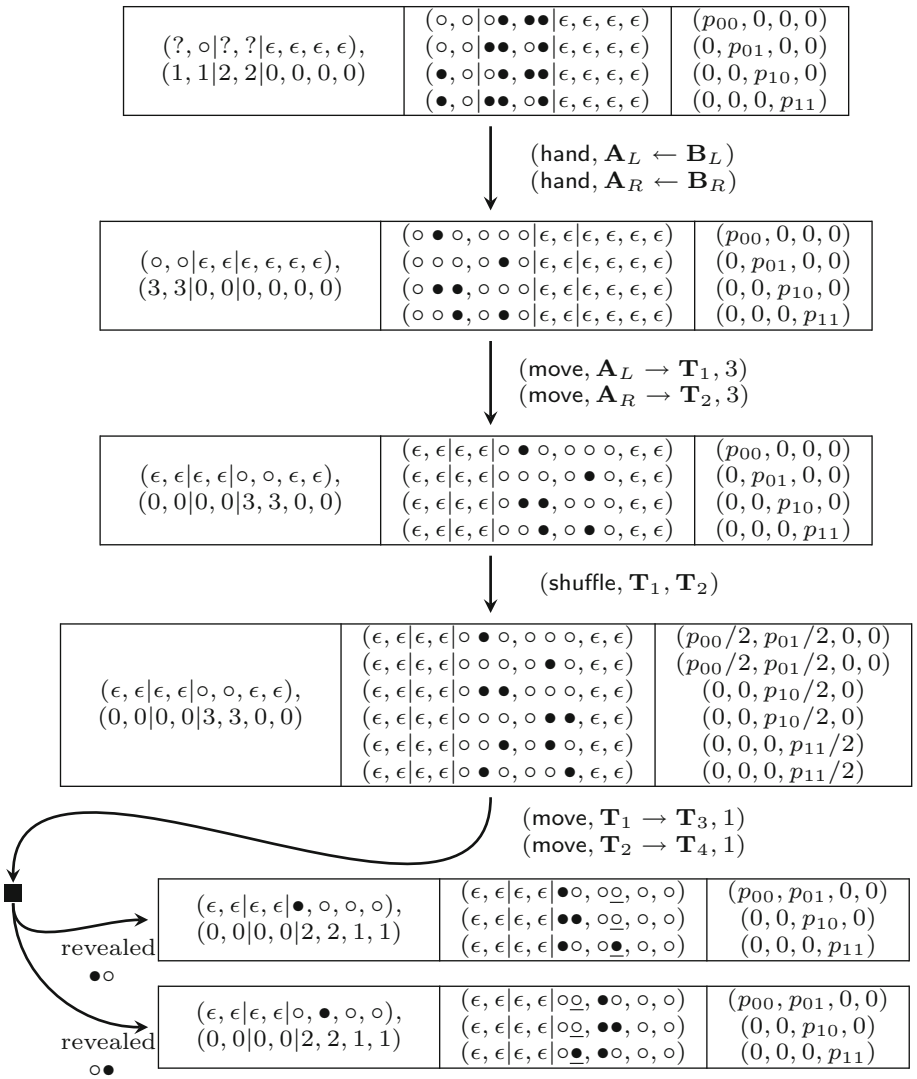

Fig. 4 Extended diagram of the coin-based AND protocol

2.5 Extended Diagram

This subsection briefly reviews the extended diagram introduced in [15] with the concrete example described in Fig. 4.

The diagram consists of nodes, and each node is connected to its neighboring node(s) through action(s). For example, Fig. 4 contains six nodes. The topmost node corresponds to an initial arrangement, and it is connected to the next node which corresponds to an arrangement after the actions of hand (Steps 1 and 2 in the six-coin AND protocol in Section 3.2).

Each node consists of a "detectable sequence," "arrangements," and "probability traces." Let us look at the topmost node in Fig. 4 again. There are four entries, each of which consists of the arrangement and the probability trace, which comes from one of the four kinds of inputs (0, 0), (0, 1), (1, 0), and (1, 1). The first entry corresponds to the input (0, 0). The arrangement is an initial one for input (0, 0) of the protocol itself, which is derived only when the input sequence is (0, 0) with a probability of p_{00}, and hence, the first coordinate of the probability trace is p_{00} and the remaining three coordinates are 0.

Similarly, let us consider the fourth node in Fig. 4, which corresponds to the state after shuffle (Step 5 in the six-coin AND protocol in Section 3.2). The probability trace in the first entry is $(p_{00}/2,\ p_{01}/2,\ 0,\ 0)$ which means that the arrangement $(\epsilon, \epsilon|\circ\bullet\circ, \circ\circ\circ|\epsilon, \epsilon)$ comes from inputs (0, 0) (shuffle does not change the arrangement) with probability $p_{00}/2$ and (0, 1) (shuffle changes one) with probability $p_{01}/2$.

Note that, if there is more than one detectable sequence after some action (move at Step 7 in the six-coin AND protocol in Section 3.2, for example), we prepare a node for each sequence as in the fifth and sixth nodes in Fig. 4.

3 Examples of the Coin-based Protocol

In this section, we give examples of coin-based protocols for NOT, AND, and COPY computations, and check their correctness and security with the extended diagram [15].

3.1 NOT Protocol

Assume that Alice holds a coin-based commitment; that is, she grabs a coin that encodes a one-bit information according to Eq. (1), as illustrated in Fig. 1. Then, the NOT computation can be executed by turning over the coin. For example, hand performs such a computation. With hand, the correctness and security trivially hold. Note that, ignoring the security, flip also performs the NOT computation.

3.2 AND Protocol

Let $a \in \{0, 1\}$ and $b \in \{0, 1\}$ be private inputs of Alice and Bob, respectively. Also, let $\overline{a}$ and $\overline{b}$ be flipped bits of a and b, namely, $a \oplus 1$ and $b \oplus 1$, respectively. Alice and Bob initially grab stacks of coins as follows: $\mathbf{a}_L$ is set to $\mathbf{a}_L = \overline{a}$ according to the encoding defined in Eq. (1), $\mathbf{a}_R$ is always $\circ$, $\mathbf{b}_L = \overline{b}\bullet$, and $\mathbf{b}_R = b\bullet$. Namely, the initial arrangement can be written as

$$\left(\overline{a}, \circ|\overline{b}\bullet, b \bullet |\epsilon, \epsilon, \epsilon, \epsilon\right).$$

Note that there are four candidates $\Gamma^{ab} \in U$ for the initial arrangement of this protocol.

Six-coin AND Protocol $\mathcal{P}^{\mathsf{AND}}_{\mathsf{coin}}$
Input:

$$\Gamma^{ab} = \begin{cases} (\circ, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (0, 0)) \\ (\circ, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (0, 1)) \\ (\bullet, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (1, 0)) \\ (\bullet, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (1, 1)) \end{cases} \tag{2}$$

Steps:

1. (hand, $\mathbf{A}_L \leftarrow \mathbf{B}_L$)
2. (hand, $\mathbf{A}_R \leftarrow \mathbf{B}_R$)
3. (move, $\mathbf{A}_L \rightarrow \mathbf{T}_1, 3$)
4. (move, $\mathbf{A}_R \rightarrow \mathbf{T}_2, 3$)
5. (shuffle, $\mathbf{T}_1, \mathbf{T}_2$)
6. (move, $\mathbf{T}_1 \rightarrow \mathbf{T}_3, 1$)
7. (move, $\mathbf{T}_2 \rightarrow \mathbf{T}_4, 1$)

Output:

$$\begin{cases} \mathsf{bottom}(\mathbf{t}_1) & ((\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2)) = (\circ, \bullet)) \\ \mathsf{bottom}(\mathbf{t}_2) & (\text{otherwise}) \end{cases} \tag{3}$$

The result of the protocol also follows the encoding defined in Eq. (1).

We can check the correctness and security of $\mathcal{P}^{\mathsf{AND}}_{\mathsf{coin}}$ by using the extended diagram. Fig. 4 shows a summary of the diagram. In this diagram, the topmost node consists of triplet of the detectable sequence, the arrangements, and the probability traces of inputs, namely, just before the first step.

We first check the correctness. The output is the bottom of the corresponding underlined coin in the final step of this figure, such as[3] $\circ\underline{\circ}$ and $\circ\underline{\bullet}$. With this diagram, it is obvious that $(a, b) = (1, 1)$, namely, the fourth component in the probability trace, p_{11}, is non-zero if and only if the output is $\mathsf{bottom}(\circ\underline{\bullet}) = \circ$ which is the encoding of 1.

We now discuss the security. If $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2))$ is $(\bullet, \circ)$ in Fig. 4, the sum of the probability traces is $(p_{00}, p_{01}, p_{10}, p_{11})$. Namely, the probability distribution of the input after the topmost coin of each stack is removed is unchanged from the viewpoint of the players and observers. This means that no information leaks through the protocol. Similarly, no information leaks when $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2))$ is $(\circ, \bullet)$. Hence, we have confirmed the security of the protocol.

3.3 COPY Protocol

Here, we present how to make an identical commitment copy from a given coin-based commitment. In addition to a coin for Alice's secret bit a, we prepare two coins of $\bullet(= 0)$ and two more dummy coins. Alice stacks these coins as $\circ 0a0\bullet$ (where we write 0 instead of $\bullet$ for convenience) and performs rflip, resulting in $\circ ra'r\bullet$ for a random bit r where $a' = a \oplus r$. If $a' = 0$, we have $a = r$, which leads to the situation where the second and fourth coins are equal to a; conversely, if $a' = 1$, they are equal to $a \oplus 1$. From these relations, we can obtain a COPY protocol as follows. In this protocol, there are two input candidates $\Gamma^a \in U$ for $a \in \{0, 1\}$.

[3] We use underlining for clarity to indicate which coin corresponds to an output.

Five-coin COPY Protocol $\mathcal{P}^{\mathsf{COPY}}_{\mathsf{coin}}$
Input:

$$\Gamma^a = \begin{cases} (\bullet, \circ\bullet|\epsilon, \epsilon|\bullet\bullet, \epsilon, \epsilon) & (a = 0) \\ (\circ, \circ\bullet|\epsilon, \epsilon|\bullet\bullet, \epsilon, \epsilon) & (a = 1) \end{cases} \tag{4}$$

Steps:

1. $(\mathsf{hand}, \mathbf{A}_L \leftarrow \mathbf{A}_R)$
2. $(\mathsf{move}, \mathbf{A}_L \rightarrow \mathbf{T}_1, 3)$
3. $(\mathsf{rflip}, \mathbf{T}_1)$
4. $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_2, 2)$
5. **if** $\mathsf{top}(\mathbf{t}_1) = \circ$,
   $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_3, 2)$
6. **else**
   (a) $(\mathsf{flip}, \mathbf{T}_1)$
   (b) $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_3, 2)$
   (c) $(\mathsf{flip}, \mathbf{T}_1)$ ($\star$*optional*)
   (d) $(\mathsf{move}, \mathbf{T}_2 \rightarrow \mathbf{T}_1, 2)$
   (e) $(\mathsf{flip}, \mathbf{T}_1)$
   (f) $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_2, 2)$

Output:

$$(\mathsf{bottom}(\mathbf{t}_2)\mathsf{bottom}(\mathbf{t}_3)) \tag{5}$$

We can check the correctness and security, similarly to the procedure for $\mathcal{P}^{\mathsf{AND}}_{\mathsf{coin}}$, regardless of whether the optional action at Step 6(c) is executed. With the optional action, the resulting stacks are $\circ\overline{a}$; whereas, without it, one of them becomes $\bullet\overline{a}$ with a probability of $\frac{1}{2}$.

Note that the above COPY protocol, with one coin for input a, uses four additional coins and obtains two coins for a. If we use $2k + 2$ coins, instead of the four coins, we can obtain $2k$ coins for a. Specifically, we replace the stacks for $\mathbf{A}_R$ and $\mathbf{T}_1$ in the inputs as follows. Instead of $\circ\bullet$ on $\mathbf{A}_R$ ($\bullet\bullet$ on $\mathbf{T}_1$, respectively), we set $\circ\circ\cdots\circ\bullet$ ($\bullet\bullet\cdots\bullet$, respectively). The above protocol ends with two stacks, with $k + 1$ coins each, on $\mathbf{T}_1$ and $\mathbf{T}_2$. With these two stacks, the bottoms k lower coins in each obtained stack are coins for a ($2k$ coins in total).

3.4 OR Protocol

We now obtain a coin-based OR protocol by combining the above AND and NOT protocols. The OR of $a, b \in \{0, 1\}$ is computed by flipping the result of $\mathcal{P}^{\mathsf{AND}}_{\mathsf{coin}}(\Gamma^{\overline{a}\overline{b}})$. The following specifically shows the protocol. Note that, in the following protocol, we change the encoding in $\Gamma_{00}, \Gamma_{01}, \Gamma_{10}$, and Γ_{11} from the above AND protocol so that we can perform $\mathcal{P}^{\mathsf{AND}}_{\mathsf{coin}}(\Gamma^{\overline{a}\overline{b}})$ without any action in advance.

Six-coin OR Protocol $\mathcal{P}^{\mathsf{OR}}_{\mathsf{coin}}$
Input:

$$\Gamma^{ab} = \begin{cases} (\bullet, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (0, 0)) \\ (\bullet, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (0, 1)) \\ (\circ, \circ|\bullet\bullet, \circ\bullet|\epsilon, \epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (1, 0)) \\ (\circ, \circ|\circ\bullet, \bullet\bullet|\epsilon, \epsilon, \epsilon, \epsilon, \epsilon) & ((a, b) = (1, 1)) \end{cases} \tag{6}$$

Steps:

1. execute $\mathcal{P}^{\mathsf{AND}}_{\mathsf{coin}}(\Gamma^{ab})$
2. **if** $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2)) = (\circ, \bullet)$
   $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_3, 2)$
3. **else if** $(\mathsf{top}(\mathbf{t}_1), \mathsf{top}(\mathbf{t}_2)) = (\bullet, \circ)$
   $(\mathsf{move}, \mathbf{T}_2 \rightarrow \mathbf{T}_3, 2)$
4. $(\mathsf{flip}, \mathbf{T}_3)$
5. $(\mathsf{move}, \mathbf{T}_3 \rightarrow \mathbf{T}_5, 2)$

Output:

$$\mathsf{bottom}(\mathbf{t}_5) \tag{7}$$

We can check the correctness and security of this protocol in the same manner as for the AND protocol.

3.5 XOR Protocol

Next, we explain a coin-based XOR protocol executed by Alice and Bob. To compute $a \oplus b$, we introduce a random bit r, which is the remainder of the number of flips in rflip divided by 2. Note that both the number of flips and r are unknown to Alice and Bob. We add this r to both a and b, that is, we have $a' = a \oplus r$ and $b' = b \oplus r$ so that we compute $a' \oplus b' = a \oplus b$.

Precisely, a' and b' are generated for an unknown unique r, as follows: We place the coin for b onto the coin for a as ba, and randomly flip the stack to $b'a' = ba$ (unchanged when $r = 0$) or $a'b' = \overline{a}\overline{b}$ (flipped when $r = 1$). Therefore, after the random flip, these coins are for $a' = a \oplus r$ and $b' = b \oplus r$. To compute $a' \oplus b' = a \oplus b$, we select one of coins for a' and b', and then flip the other coin if the selected coin is for a bit one. To avoid information leakage, we need to cover the coins with dummy coins and shuffle the coins in the selection.

The following shows the protocol. The initial arrangement is $(\circ a\bullet, \epsilon| \circ \overline{b}\bullet, \epsilon|\epsilon, \epsilon, \epsilon)$, and similarly to the above two protocols, there are four input candidates $\Gamma^{ab} \in U$ for this protocol.

Six-coin XOR Protocol $\mathcal{P}_{\mathsf{coin}}^{\mathsf{XOR}}$
Input:

$$\Gamma^{ab} = \begin{cases} (\circ\bullet\bullet, \epsilon | \circ\circ\bullet, \epsilon | \epsilon, \epsilon, \epsilon) & ((a,b) = (0,0)) \\ (\circ\bullet\bullet, \epsilon | \circ\bullet\bullet, \epsilon | \epsilon, \epsilon, \epsilon) & ((a,b) = (0,1)) \\ (\circ\circ\bullet, \epsilon | \circ\circ\bullet, \epsilon | \epsilon, \epsilon, \epsilon) & ((a,b) = (1,0)) \\ (\circ\circ\bullet, \epsilon | \circ\bullet\bullet, \epsilon | \epsilon, \epsilon, \epsilon) & ((a,b) = (1,1)) \end{cases} \tag{8}$$

Steps:

1. $(\mathsf{hand}, \mathbf{A}_L \leftarrow \mathbf{B}_L)$
2. $(\mathsf{move}, \mathbf{A}_L \rightarrow \mathbf{T}_1, 6)$
3. $(\mathsf{rflip}, \mathbf{T}_1)$
4. $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_2, 3)$
5. $(\mathsf{shuffle}, \mathbf{T}_1, \mathbf{T}_2)$
6. $(\mathsf{move}, \mathbf{T}_1 \rightarrow \mathbf{T}_3, 1)$
7. **if** $\mathsf{top}(\mathbf{t}_1) = \bullet$
   (a) $(\mathsf{flip}, \mathbf{T}_2)$
   (b) $(\mathsf{move}, \mathbf{T}_2 \rightarrow \mathbf{T}_3, 2)$
8. **else**
   $(\mathsf{move}, \mathbf{T}_2 \rightarrow \mathbf{T}_3, 2)$

Output:

$$\mathsf{bottom}(\mathbf{t}_3) \tag{9}$$

In a manner similar to that of previous protocols, we can check the correctness and security.

4 Toward More Protocols

The previous section shows concrete examples of coin-based protocols. Let us discuss protocols for other functionalities. Note that the set {NOT, AND} is known to be functionally complete. Hence, a protocol for any (two-variable) Boolean function can be realized by combining the coin-based NOT and AND protocols presented above.

Let us consider the composition of logical gates in a logic circuit. If the fan-out of a logical gate is two or more, the signal is duplicated to be connected to each gate. In a coin-based protocol, a one-bit value can be duplicated by the coin-based COPY protocol presented in Section 3.3. Therefore, by combining the coin-based NOT, AND, and COPY protocols, a protocol for any function can be constructed.

To make the discussion on the composition more clear and universal, we introduce another action, pick, and modify the initial states of the coin-based protocols presented in the above.

4.1 Pickup Action

Our example protocols in Section 3 produce their output as a coin with an unknown state (head or tail) because the coin is covered by another coin. Hence, it may be possible to compose protocols if a player can move the resulting coin of the first protocol into her or his hand as an input to the next protocol.

Therefore, if it is possible to pick the resulting coin in one's hand without revealing its information, any secure function evaluation can be executed with our examples of NOT, AND, and COPY protocols. To this end, we provide the following action, called pick, which can also be implemented by humans.

- $(\mathsf{pick}, \mathbf{P}_1 \rightarrow \mathbf{P}_2, n)$ for $\mathbf{P}_1 \in \{\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_k\}$ and $\mathbf{P}_2 \in \{\mathbf{A}_L, \mathbf{A}_R, \mathbf{B}_L, \mathbf{B}_R\}$ with $\mathbf{p}_2 = \epsilon$: A player, Alice, places her empty hand, say her left hand, over the stack of m coins $\mathbf{p}_1 \in \mathcal{S}^k$ on the table area $\mathbf{P}_1$ to hide the stack. Then, she slips her right hand under her left hand, picks the upper n coins from the stack under her left hand, and moves the stack of n coins to the table so that only the topmost coin is visible. After that, she slips her right hand under her left hand again, lifts the remaining coins, and closes her left hand to hold the raised coins without revealing the surfaces of the coins. This action changes the tuple from $(\cdots, \mathbf{p}_2 = \epsilon, \cdots | \cdots, \mathbf{p}_1, \cdots)$ to $(\cdots, \mathsf{turn}(\mathbf{p}_1^{(l)}), \cdots | \cdots, \mathbf{p}_1^{(u)}, \cdots)$, where $\mathbf{p}_1^{(u)}$ and $\mathbf{p}_1^{(l)}$ are the stacks of upper n coins and lower $m-n$ coins of $\mathbf{p}_1$, respectively.

The following is an implementation of pick that picks up a coin in three steps. Assume that there is a stack of two coins $\mathbf{t}_1$ ($\circ\bullet$ or $\circ\circ$ if the result is 1 or 0, respectively) on the table area $\mathbf{T}_1$ as a result of the AND protocol, and that a player, Alice, wants to pick up the lower coin by performing $(\mathsf{pick}, \mathbf{T}_1 \rightarrow \mathbf{A}_L, 1)$ with her left hand $\mathbf{A}_L$ where $\mathbf{a}_L = \epsilon$. She places her hand over the stack to hide it (left of Fig. 5), removes the upper coin(s) of $\mathbf{t}_1^{(u)}$ by picking the coin(s) with her right hand (middle), and holds the lower coin(s) $\mathbf{t}_1^{(l)}$ in her left hand by slightly lifting the coin(s) with her right hand (right). Note that, during the last step, the direction of the coin or coin stack $\mathbf{t}_1^{(l)}$ is changed to $\mathsf{turn}(\mathbf{t}_1^{(l)})$ because the top of $\mathbf{t}_1^{(l)}$ contacts Alice's left palm (which is the bottom side in $\mathbf{A}_L$).

4.2 Modified Protocols Suitable for Composition

We are now ready to modify the protocols in Section 3. In Section 3, for simplicity, we assume that the players initially hold coins in their hands, in accordance with their inputs. In this subsection, we modify the initial arrangement of the protocol so that the players do not hold coins but the coins are initially placed on the table.

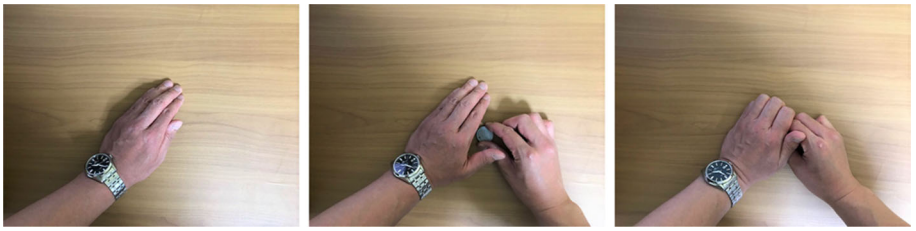

Fig. 5 Example of picking the resulting coin

Let us first recall the COPY protocol presented in Section 3.3, with which the initial arrangement can be written as $(a, \circ\bullet|\epsilon, \epsilon|\bullet, \bullet, \epsilon, \epsilon)$. Instead of this, let us consider the arrangement $(\epsilon, \epsilon|\epsilon, \epsilon|\bullet, \bullet, \circ\overline{a}, \circ\bullet)$ where the coin corresponding to $\overline{a}$ is put on the table area $\mathbf{T}_3$ with a dummy coin (and both Alice's hands are empty). Using the pick action, we can easily move this commitment on $\mathbf{T}_3$ to Alice's hand $\mathbf{A}_L$, namely $(\mathsf{pick}, \mathbf{T}_3 \to \mathbf{A}_L, 1)$. Then, $(\mathsf{move}, \mathbf{T}_4 \to \mathbf{A}_R, 2)$ brings the initial arrangement for the original COPY protocol. Thus, we modify the COPY protocol to start with a commitment to a on the table. Note that its output, namely two coins with dummy coins corresponding to a are on the table.

Next, let us recall the coin-based AND protocol in Section 3.2. The initial arrangement was $(\overline{a}, \circ|\overline{b}\bullet, b\bullet|\epsilon, \epsilon, \epsilon, \epsilon)$ for their inputs (a, b). We change the initial arrangement to

$$(\epsilon, \epsilon|\epsilon, \epsilon|\circ\overline{a}, \bullet, \circ\overline{b}, \epsilon).$$

Then, using the modified COPY protocol together with other actions, we can easily transform the initial arrangement into $(\epsilon, \epsilon|\epsilon, \epsilon|\circ a, \bullet, \circ b, \circ\overline{b})$. After this, the following actions create coin-based commitments in hands that suffice for the AND computation.

1 $(\mathsf{pick}, \mathbf{T}_1 \to \mathbf{A}_L, 1)$
2 $(\mathsf{pick}, \mathbf{T}_2 \to \mathbf{A}_R, 1)$
3 $(\mathsf{pick}, \mathbf{T}_3 \to \mathbf{B}_L, 2)$
4 $(\mathsf{pick}, \mathbf{T}_4 \to \mathbf{B}_R, 2)$

That is, after these actions, the arrangement becomes identical to the initial arrangement of the coin-based AND protocol in Section 3.2; therefore, the players compute $a \wedge b$ by performing Steps 1 to 9 of the protocol.

Note that it may be difficult to execute the above pick actions only by two players. In such case, another player, Charlie, provides assistance. For example, the first action is proceeded by:

a. $(\mathsf{pick}, \mathbf{T}_1 \to \mathbf{C}_L, 1)$
b. $(\mathsf{hand}, \mathbf{C}_R \leftarrow \mathbf{C}_L)$
c. $(\mathsf{hand}, \mathbf{A}_L \leftarrow \mathbf{C}_R)$,

where $\mathbf{C}_L$ and $\mathbf{C}_R$ are Charlie's left hand and right hand, respectively. Hereinafter, we simply write that the pick actions are executed by Alice and Bob.

Similarly, one can easily modify the protocols for OR and XOR, where the coins are on the table in the initial arrangement.

4.3 Examples of Composition Protocols

In this section, we discuss a three-input AND protocol and a three-input majority decision protocol for inputs $(a, b, c) \in \{0, 1\}^3$, as examples of composition.

The three-input AND is evaluated by $a \wedge b \wedge c = (a \wedge b) \wedge c$, which executes the (two-input) AND protocol twice. The procedure of the three-input coin-based AND protocol is as follows:

1 Alice, Bob, and Charlie place their inputs $\circ a$, $\circ b$, and $\circ c$ on table $\mathbf{T}_{1,1}$, $\mathbf{T}_{2,1}$, and $\mathbf{T}_{3,1}$, respectively.
2 Alice and Bob perform the modified coin-based AND protocol so that they obtain $\mathbf{t}_{1,1} = \circ(\overline{a \wedge b})$, namely the stack of resulting coins on $\mathbf{T}_{1,1}$ for $a \wedge b$.
3 Alice and Charlie perform the modified coin-based AND protocol to compute $(a \wedge b) \wedge c$ as in Step 2.

The three-input majority decision is evaluated by $(a \wedge b) \vee (b \wedge c) \vee (c \wedge a)$. The coin-based OR protocol in Section 3.4 is a natural composition of the coin-based NOT protocol and AND protocol. Therefore, we use the protocols for NOT, AND, COPY, and OR as building blocks below.

1 Alice, Bob, and Charlie place their inputs $\circ\overline{a}$, $\circ\overline{b}$, and $\circ\overline{c}$ on table $\mathbf{T}_{1,1}$, $\mathbf{T}_{2,1}$, and $\mathbf{T}_{3,1}$, respectively.
2 Alice copies her input by using the modified COPY protocol so that the resulting stacks are $\mathbf{t}_{1,1} = \mathbf{t}_{1,2} = \circ\overline{a}$.
3 Bob and Charlie each copy their inputs as in Step 2 above. The resulting stacks are $\mathbf{t}_{2,1} = \mathbf{t}_{2,2} = \circ\overline{b}$ and $\mathbf{t}_{3,1} = \mathbf{t}_{3,2} = \circ\overline{c}$.
4 Alice and Bob perform the modified coin-based AND protocol with coin stacks $\mathbf{t}_{1,1}$ and $\mathbf{t}_{2,1}$ to compute $a \wedge b$. Let $\mathbf{t}_{1,1} = \circ(\overline{a \wedge b})$ be the resulting stack of coins.
5 Bob and Charlie perform the modified coin-based AND protocol with coin stacks $\mathbf{t}_{2,2}$ and $\mathbf{t}_{3,1}$ to compute $b \wedge c$. Let $\mathbf{t}_{2,1} = \circ(\overline{b \wedge c})$ be the resulting stack of coins.

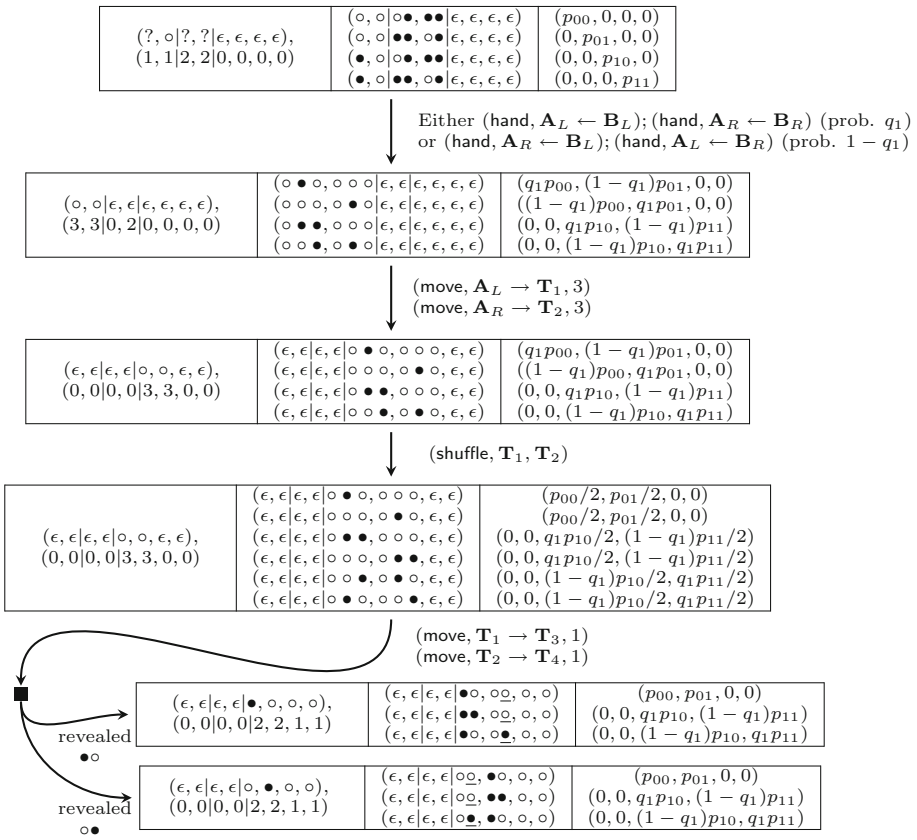

Fig. 6 Extended diagram of the coin-based AND protocol with erroneous hand actions

6 Charlie and Alice perform the modified coin-based AND protocol with coin stacks $\mathbf{t}_{3,1}$ and $\mathbf{t}_{1,2}$ to compute $c \wedge a$. Let $\mathbf{t}_{3,1} = \circ(\overline{c \wedge a})$ be the resulting stack of coins.

7 Alice and Bob perform the modified coin-based OR protocol for $\mathbf{t}_{1,1} = \circ(\overline{a \wedge b})$ and $\mathbf{t}_{2,1} = \circ(\overline{b \wedge c})$ to obtain $\mathbf{t}_{1,1} = \circ(\overline{(a \wedge b) \vee (b \wedge c)})$, namely the resulting stack of coins for $(a \wedge b) \vee (b \wedge c)$.

8 Alice and Charlie perform the modified OR protocol, as in the previous step, to compute $((a \wedge b) \vee (b \wedge c)) \vee (c \wedge a)$, with $\mathbf{t}_{1,1} = \circ(\overline{(a \wedge b) \vee (b \wedge c)})$ and $\mathbf{t}_{3,1} = \circ(\overline{c \wedge a})$.

Similarly to the above compositions, a protocol for any function can be constructed with the coin-based protocols for NOT, AND, and COPY.

5 Effect of Human Error

In Section 3, we propose coin-based protocols that produce correct results when the players perform them. However, because humans make mistakes in general, the effect of human error on the protocols is discussed. For example, for the card-based AND protocol, Mizuki and Komano discussed the effect, *the information leakage*, human error [15]. Thus, in this section, we analyze the effect of human error in terms of information leakage and correctness.

Let us discuss the effect on the six-coin AND protocol $\mathcal{P}^{\mathsf{AND}}_{\mathsf{Coin}}$ in Section 3.2. This protocol consists of the following five actions; "set the initial configuration," "hand coins from Bob's left hand (right hand, respectively) to Alice's left hand (right hand, respectively)," "move the stack of coins on the table," "shuffle the stacks of coins on the table," and "move the topmost coins of the stacks." Among these actions, Alice and Bob may make a mistake when they "hand coins from Bob's hand to Alice's hand"; namely, Bob may hand the stack of coins in his left hand (right hand, respectively) to Alice's *right* hand (*left* hand, respectively) incorrectly. As for the other four actions, because these actions are simple to execute, it seems that mistakes while performing them would be rare. Hence, let us consider the erroneous six-coin AND protocol where the following two steps are executed, with the probability of $(1 - q_1)$, instead of Steps 1 and 2 in $\mathcal{P}^{\mathsf{AND}}_{\mathsf{Coin}}$.

$(1')$ $(\mathsf{hand}, \mathbf{A}_L \leftarrow \mathbf{B}_R)$
$(2')$ $(\mathsf{hand}, \mathbf{A}_R \leftarrow \mathbf{B}_L)$

Figure 6 shows the extended diagram for the incorrectly performed protocol. If Bob hands a stack of coins to the

wrong hand as (1') and (2') above, the protocol outputs $a \wedge \bar{b}$ instead of $a \wedge b$.

Each sum of the probability traces of the bottom two nodes (final states) in Fig. 6 is $(p_{00}, p_{01}, p_{10}, p_{11})$, which means that *no* information on the inputs leaks even if the players make a mistake during the hand action. However, as we discussed the erroneous protocol outputs an incorrect result. For example, the protocol outputs 0 with an input pair (1, 1) with the probability of $(1 - q_1)p_{11}$ as in the second line from the bottom of the bottommost node in the diagram. Conversely, the protocol outputs 1 with an input pair (1, 0) with the probability of $q_1 p_{10}$, as in the first line from the bottom of the bottommost node in the diagram.

6 Conclusions

This paper introduced the formal treatment for coin-based protocols and presented concrete examples to show that the secure multi-party computation is reliably executed with physical coins. An intriguing future work includes the development of more practical protocols with fewer coins.

Acknowledgements This work was supported by JSPS KAKENHI Grant Numbers JP17K00001 and JP18H05289. We would like to thank the reviewers for their invaluable comments.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Abe, Y., Iwamoto, M., Ohta, K.: Efficient private PEZ protocols for symmetric functions. In: D. Hofheinz, A. Rosen (eds.) Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 11891, pp. 372–392. Springer (2019). doi:https://doi.org/10.1007/978-3-030-36030-6_15
2. Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. Theor. Comput. Sci. **306**(1–3), 69–84 (2003). https://doi.org/10.1016/S0304-3975(03)00210-X
3. den Boer, B.: More efficient match-making and satisfiability: *The Five Card Trick*. In: J. Quisquater, J. Vandewalle (eds.) Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings, *Lecture Notes in Computer Science*, vol. 434, pp. 208–217. Springer (1989). doi:https://doi.org/10.1007/3-540-46885-4_23
4. Crépeau, C., Kilian, J.: Discreet solitary games. In: D.R. Stinson (ed.) Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings, *Lecture Notes in Computer Science*, vol. 773, pp. 319–330. Springer (1993). doi:https://doi.org/10.1007/3-540-48329-2_27
5. D'Arco, P., Prisco, R.D.: Secure computation without computers. Theor. Comput. Sci. **651**, 11–36 (2016). https://doi.org/10.1016/j.tcs.2016.08.003
6. Goldwasser, S.: Multi-party computations: Past and present. In: J.E. Burns, H. Attiya (eds.) Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997, pp. 1–6. ACM (1997). doi:https://doi.org/10.1145/259380.259405
7. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: T. Iwata, J.H. Cheon (eds.) Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I, *Lecture Notes in Computer Science*, vol. 9452, pp. 783–807. Springer (2015). doi:https://doi.org/10.1007/978-3-662-48797-6_32
8. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: N. Koblitz (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings, *Lecture Notes in Computer Science*, vol. 1109, pp. 104–113. Springer (1996). doi:https://doi.org/10.1007/3-540-68697-5_9
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: M.J. Wiener (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings, *Lecture Notes in Computer Science*, vol. 1666, pp. 388–397. Springer (1999). doi:https://doi.org/10.1007/3-540-48405-1_25
10. Komano, Y., Mizuki, T.: Multi-party computation based on physical coins. In: D. Fagan, C. Martín-Vide, M. O'Neill, M.A. Vega-Rodríguez (eds.) Theory and Practice of Natural Computing - 7th International Conference, TPNC 2018, Dublin, Ireland, December 12-14, 2018, Proceedings, *Lecture Notes in Computer Science*, vol. 11324, pp. 87–98. Springer (2018). doi:https://doi.org/10.1007/978-3-030-04070-3_7
11. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
12. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. Cryptology ePrint Archive. https://eprint.iacr.org/2015/1031 (2015). Accessed 15 May 2020
13. Miyahara, D., Komano, Y., Mizuki, T., Sone, H.: Cooking cryptographers: secure multiparty computation based on balls and bags. In: 34th IEEE 34th Computer Security Foundations Symposium (CSF), Dubrovnik, Croatia. IEEE (2021). https://doi.org/10.1109/CSF51468.2021.00034

14. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theor. Comput. Sci. **622**, 34–44 (2016). https://doi.org/10.1016/j.tcs.2016.01.039
15. Mizuki, T., Komano, Y.: Analysis of information leakage due to operative errors in card-based protocols. In: C.S. Iliopoulos, H.W. Leong, W. Sung (eds.) Combinatorial Algorithms - 29th International Workshop, IWOCA 2018, Singapore, July 16-19, 2018, Proceedings, *Lecture Notes in Computer Science*, vol. 10979, pp. 250–262. Springer (2018). doi:https://doi.org/10.1007/978-3-319-94667-2_21
16. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: X. Wang, K. Sako (eds.) Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings, *Lecture Notes in Computer Science*, vol. 7658, pp. 598–606. Springer (2012). doi:https://doi.org/10.1007/978-3-642-34961-4_36
17. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. Int. J. Inf. Sec. **13**(1), 15–23 (2014). https://doi.org/10.1007/s10207-013-0219-4
18. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **100-A**(1), 3–11 (2017). doi:https://doi.org/10.1587/transfun.E100.A.3
19. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: X. Deng, J.E. Hopcroft, J. Xue (eds.) Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009. Proceedings, *Lecture Notes in Computer Science*, vol. 5598, pp. 358–369. Springer (2009). doi:https://doi.org/10.1007/978-3-642-02270-8_36
20. Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. Australas. J Comb. **36**, 279–294 (2006)
21. Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. Theor. Comput. Sci. **411**(10), 1283–1310 (2010). https://doi.org/10.1016/j.tcs.2009.10.023
22. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Comput. **22**(2), 361–371 (2018). https://doi.org/10.1007/s00500-017-2858-2
23. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. Int. J. Inf. Sec. **19**(4), 445–452 (2020). https://doi.org/10.1007/s10207-019-00463-w