Public-PEZ Cryptography*

Soma Murata[1], Daiki Miyahara[1,3], Takaaki Mizuki[2], and Hideaki Sone[2]

[1] Graduate School of Information Sciences, Tohoku University
6–3–09 Aramaki-Aza-Aoba, Aoba-ku, Sendai, 980–8578, Japan
[2] Cyberscience Center, Tohoku University
6–3 Aramaki-Aza-Aoba, Aoba-ku, Sendai, 980–8578, Japan
[3] National Institute of Advanced Industrial Science and Technology
2–3–26, Aomi, Koto-ku Tokyo, 135-0064, Japan

Abstract. Secure multiparty computation (MPC) is a cryptographic technique that enables us to evaluate a predetermined function over players' private inputs while hiding information about the inputs. MPC can be conducted using a "private PEZ protocol," that uses PEZ candies and a dispenser. Specifically, in a private PEZ protocol, players first fill a predetermined sequence of candies in a dispenser. Then, each player in turn privately pops out a number of candies, wherein the number depends on their private input (without anybody else knowing how many candies pop out). The next candy to be popped out of the dispenser indicates the output value of the function. Thus, private PEZ protocols are fun and useful. One drawback would be that every player must pop out candies from the dispenser secretly, implying that a private PEZ protocol is vulnerable to dishonest players, for example, a player could peep the candies inside the dispenser. To overcome this drawback, we herein propose MPC protocols that do not need private actions such as secretly popping out candies after the setup (although each player rearranges the candies secretly in a setup phase, any illegal actions can be caught). That is, we construct a computational model of "public-PEZ cryptography," where any protocol within the model can be publicly executed. Especially, the proposed public-PEZ AND protocol, which uses only five candies and two dispensers, is simple and easy for conducting a secure computation of the AND function.

Keywords: Secure Multiparty Computations · Recreational Cryptography · Private PEZ Protocols · Card-based Cryptography

1 Introduction

1.1 Background

Secure multiparty computation (MPC) is a cryptographic technique for evaluating a predetermined function over players' private inputs while hiding information

* This paper appears in Proceedings of ISC 2020. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-62974-8_4.

Fig. 1: PEZ candies, packages, and a dispenser.

Fig. 2: A deck of cards.

about the inputs. Interestingly, MPC can be performed using not only computers but also everyday objects. Some examples are *private PEZ protocols* using PEZ candies and a dispenser (as shown in Figure 1) and *card-based protocols* using a deck of physical cards (as shown in Figure 2). Using such physical cryptographic protocols, we can visually understand what MPC is and how secure the protocols are. Thus, they attract not only cryptographers but also non-experts, such as high school students, and they can be effectively used as educational tools.

Private PEZ protocols were first introduced in 2003 by Balogh *et al.* [2], and their results were improved recently by Abe *et al.* [1] in 2019. In a private PEZ protocol, players first fill in a dispenser with a predetermined sequence of candies whose order depends on the function that they want to securely compute. Subsequently, each player privately pops out a number of candies such that the number depends on their private input. Finally, the remaining topmost candy left in the dispenser becomes the output of the function. Thus, private PEZ protocols are fun and useful. One drawback is that every player must take out candies from the dispenser secretly, implying that a private PEZ protocol is vulnerable to dishonest players. For example, when a player pops out candies secretly, they could maliciously peep the candies inside the dispenser or replace them with another sequence of candies as per their preference.

1.2 Contributions

In this paper, to overcome the drawback of the above-mentioned private PEZ protocols, which require players' private actions, we consider a new usage of PEZ candies and dispensers by borrowing the ideas behind card-based protocols. That is, we design novel PEZ protocols that can be publicly executed. Specifically, we first propose a secure AND protocol using five PEZ candies and two dispensers; it allows Alice and Bob to compute the AND value of their private inputs without revealing them. Carrying the idea behind this protocol further, we construct a computational model of *public-PEZ cryptography*. Following this model, we present the formal description of our AND protocol. We also discuss some implementation issues.

1.3 Related Work

As mentioned above, we borrow the ideas and techniques from card-based protocols. The first card-based protocol was proposed by den Boer [5] in 1989; his famous protocol called the "five-card trick" performs a secure computation of the AND function. Our constructions are inspired by the card-based AND protocols [16,21] that use a shuffling operation called a "random bisection cut." In card-based protocols, all players first place a pair of cards face-down on a table whose order depends on their private input. Subsequently, they perform MPC by publicly shuffling and turning over the sequence of cards to obtain the output. A formal computational model of card-based protocols was presented (and was then reviewed) in the literature [11,19,20,31]. Based on the computational model, Koch *et al.* [11], Francis *et al.* [7], Kastner *et al.* [9], and Koch *et al.* [10] provided tight lower bounds on the number of required cards. In addition to simple MPCs, there are card-based protocols for zero-knowledge proof [3,6,12,15,28,29] and secure comparison [13,14,30]. Furthermore, there is another direction of card-based cryptography that relies on private actions [24–27,33]. Protocols using other everyday objects have been proposed, such as those using a dial lock [17], the 15-puzzle [18], envelopes [8], tamper-evident seals [23], and a visual secret sharing scheme [4].

1.4 Outline

The remainder of this paper is organized as follows. In Section 2, we present a simple and easy-to-implement AND protocol using five PEZ candies and two dispensers. In Section 3, we define each operation appearing in public-PEZ protocols and obtain a computational model. In Section 4, we formally describe the AND protocol introduced in Section 2 based on the model shown in Section 3. In Section 5, we discuss the feasibility of implementing the shuffling operations of public-PEZ protocols. The concluding statements are presented in Section 6.

2 Public-PEZ AND Protocol

In this section, we present a simple and easy-to-implement AND protocol using five PEZ candies and two dispensers.

Assume that Alice and Bob hold private input bits $a \in \{0,1\}$ and $b \in \{0,1\}$, respectively. They want to learn $a \wedge b$, namely the AND value of their inputs, without revealing the input values more than necessary. They only require two packages of PEZ candies (of different flavors) and two identical dispensers[4]. We assume that all candies are indistinguishable in terms of appearance, *i.e.*, they have the same color and shape; the PEZ candies sold in Japan satisfy this condition, *i.e.*, the lemon and orange candies appear identical, as shown in

[4] Two identical dispensers can be easily obtained by buying two sets of the same product.

Fig. 3: The PEZ candies sold in Japan; the lemon and orange candies are all white and indistinguishable.

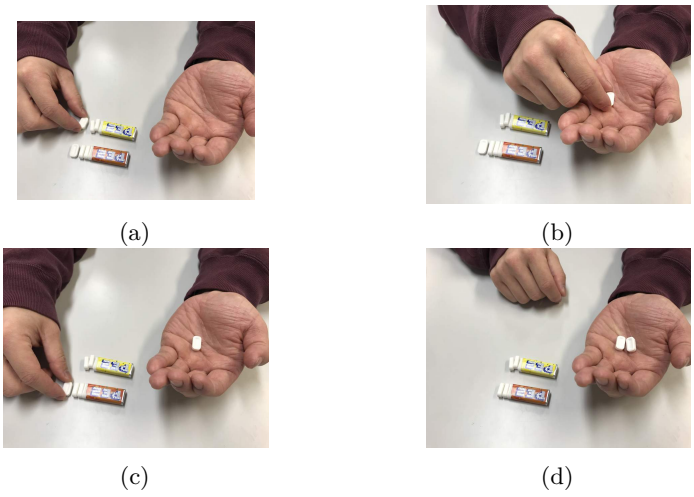


Fig. 4: How to prepare two candies of different flavors on Alice's palm.

Figure 3. We also assume that one cannot distinguish two candies of different flavors by their smells[5].

Let us consider how a Boolean value can be encoded by a flavor of the candy: Let the lemon flavor be denoted by 0, and let the orange flavor be denoted by 1. As shown in Figure 3, candies with the same flavor are packed in the same package. Let Alice take a lemon candy and an orange candy from the packages publicly and place the two candies with different flavors on her palm, as shown in Figure 4. Next, let Alice arrange the two candies inside her hand (without Bob knowing their order), as shown in Figure 5, according to her input bit $a \in \{0, 1\}$, as follows. If $a = 0$, she rearranges the two candies in the order of lemon, and then, orange, *i.e.*, 0 to 1; otherwise, in the order of orange to lemon, *i.e.*, 1 to 0. Subsequently, she takes the two candies from her palm and places them on the

[5] This holds true at least for the authors' sense of smell.

Fig. 5: Rearrange the two candies in Alice's hand without Bob knowing its order.

(a)

(b)

(c)

(d)

Fig. 6: Place the candies (whose order matches Alice's private bit) on the table without Bob knowing the order.

table so that the order satisfies

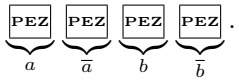
$$\underbrace{\boxed{\text{PEZ}}}_{a}\ \underbrace{\boxed{\text{PEZ}}}_{\overline{a}},$$

as shown in Figure 6. In this manner, Alice can prepare two candies (of different flavors) corresponding to her private bit $a \in \{0, 1\}$ and its negation $\overline{a}$ without Bob knowing its value. Note that Alice and Bob face each other, and Bob watches Alice's behavior during the entire process. Therefore, Alice has no choice but to place two candies of different flavors on the table; if Alice acts maliciously, such an illegal action will be always identified.

We are now ready to present our protocol; it is based on the idea behind the card-based AND protocol proposed in [16]. Our protocol proceeds as follows.

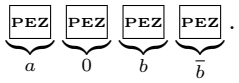
1. As described above, Alice prepares two candies corresponding to a and $\overline{a}$. Similarly, Bob prepares two candies that correspond to b and $\overline{b}$. A sequence

of candies is arranged on a table as follows:

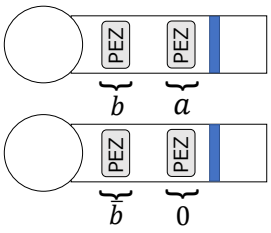


Remember that the candies are indistinguishable from each other; for example, Alice does not know how the third candy tastes (unless she bites it to confirm its flavor).

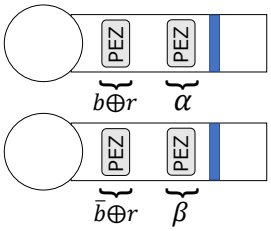
2. Alice replaces the candy corresponding to $\overline{a}$ with a candy corresponding to 0, *i.e.*, she discards (or eats) the candy corresponding to $\overline{a}$ and places a candy there corresponding to 0, which is taken from the lemon package:



3. Alice and Bob fill the four candies into two identical dispensers as follows.



4. Shuffle the two dispensers. The resulting state can be described as follows.



Here, $r \in \{0, 1\}$ is a uniformly distributed random bit, and α and β are defined as follows: if $r = 0$, $(\alpha, \beta) = (a, 0)$; if $r = 1$, $(\alpha, \beta) = (0, a)$. That is, $r = 0$ indicates that the shuffle has not swapped the two dispensers, and $r = 1$ means that the shuffle has swapped them. Moreover, if $b \oplus r = 0$, we have $a \wedge b = a \wedge r = \beta$ because $a \wedge 0 = 0$ and $a \wedge 1 = a$; if $b \oplus r = 1$, we have $a \wedge b = a \wedge \overline{r} = \alpha$ because $a \wedge \overline{0} = a$ and $a \wedge \overline{1} = 0$.

5. Pop out the candy from each dispenser and then bite them[6]. If the candy $b \oplus r$ popped out of the top dispenser corresponds to 0, the candy remaining in the bottom dispenser β represents the output $a \wedge b$; if $b \oplus r$ corresponds to 1, the candy remaining in the top dispenser α represents the output $a \wedge b$.

This is our AND protocol, using five candies and two dispensers along with one shuffle. Note that the shuffle and popping out candies can be done publicly. It should be also noted that in Step 1, hidden operations shown in Figure 5 are required to prepare players' private inputs; this is inevitable for MPC and, as mentioned before, both players cannot place any pair of candies that deviates from the encoding rule.

After Step 5, the players obtain a candy corresponding to $a \wedge b$:

$$\underbrace{\boxed{\text{PEZ}}}_{a \wedge b}.$$

If they bite it, they can know the value of $a \wedge b$. Instead, this candy can be used as an input to another computation: that is, if Carol prepares two candies according to her private bit $c \in \{0, 1\}$, starting the AND protocol again with

$$\underbrace{\boxed{\text{PEZ}}}_{a \wedge b} \underbrace{\boxed{\text{PEZ}}}_{0} \underbrace{\boxed{\text{PEZ}}}_{c} \underbrace{\boxed{\text{PEZ}}}_{\overline{c}}$$

generates a candy corresponding to $a \wedge b \wedge c$:

$$\underbrace{\boxed{\text{PEZ}}}_{a \wedge b \wedge c}.$$

Thus, a secure AND computation of more than two inputs can be easily performed.

3 Formalizing Public-PEZ Protocols

In this section, by elaborating the idea behind our AND protocol shown in Section 2, we formally define each operation on PEZ candies and dispensers and provide a computational model of public-PEZ protocols that publicly perform MPC using PEZ candies and dispensers. We borrow the ideas and terms from the computational model of card-based protocols [19].

3.1 Sequence of Candies

There are various flavors of PEZ candies, such as lemon and orange; however, as assumed before, all candies have the same appearance so that they are indistinguishable (unless they are eaten). Considering all available candies, we denote the multiset of them by $\mathcal{B}$, which we call a *box*. Any element $c \in \mathcal{B}$

[6] Alice and Bob may want to bite the candy simultaneously after splitting it for the purpose of preventing anyone from lying about the flavor.

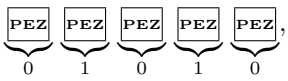
represents a flavor, such as $c \in [\texttt{lemon}, \texttt{lemon}, \texttt{orange}, \texttt{orange}, \texttt{grape}, \cdots]$. For simplicity, hereinafter, we consider only two flavors and denote them by 0 or 1. Therefore, for instance, the AND protocol presented in Section 2 works on the box

$$\mathcal{B} = [\,0, 0, 0, 1, 1\,] = [\,3 \cdot 0, 2 \cdot 1\,]$$

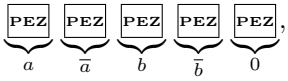
because it uses three lemon candies and two orange candies. Generally, if a protocol requires k candies corresponding to 0 and ℓ candies corresponding to 1, the box is expressed as

$$\mathcal{B} = [\,k \cdot 0, \ell \cdot 1\,].$$

Next, we consider a "sequence" of candies. When a player takes a candy from a package, its flavor is publicly known. For instance, if there are three lemon candies and two orange candies taken from the packages, the order of which is



we write this sequence as $(0, 1, 0, 1, 0)$. Given such a sequence $(0, 1, 0, 1, 0)$, let both Alice and Bob (holding input bits a and b, respectively) rearrange two candies inside their hand as in the AND protocol, shown in Section 2; then, we have



where the flavors of the left-most four candies become publicly unknown. If the flavor of candy $c \in \mathcal{B}$ is unknown to the public, we denote it by $\frac{?}{c}$. Therefore, for example, when $a = 1$ and $b = 0$, the sequence above can be written as

$$\left(\frac{?}{1}, \frac{?}{0}, \frac{?}{0}, \frac{?}{1}, 0\right).$$

Now, consider a situation where players eat the first candy; then, the flavor, 1, becomes public while the candy has disappeared. We use expression $\frac{c}{\epsilon}$ with $c \in \mathcal{B}$ to represent a candy whose flavor is known to the public, but the candy itself does not exist. Therefore, the resulting sequence (from eating the left-most candy) can be written as

$$\left(\frac{1}{\epsilon}, \frac{?}{0}, \frac{?}{0}, \frac{?}{1}, 0\right).$$

For $c \in \mathcal{B}$, we define $\mathsf{atom}(c) = \mathsf{atom}(\frac{?}{c}) = \mathsf{atom}(\frac{c}{\epsilon}) = c$. For example, $\mathsf{atom}(1) = 1$, $\mathsf{atom}(\frac{?}{0}) = 0$, and $\mathsf{atom}(\frac{1}{\epsilon}) = 1$. For the box $\mathcal{B}$ with $|\mathcal{B}| = m$, we call $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ a *sequence* from the box $\mathcal{B}$ if $\alpha_i \in \{0, 1, \frac{?}{0}, \frac{?}{1}, \frac{0}{\epsilon}, \frac{1}{\epsilon}\}$ for every i, $1 \leq i \leq m$, and $[\mathsf{atom}(\alpha_1), \mathsf{atom}(\alpha_2), \ldots, \mathsf{atom}(\alpha_m)] = \mathcal{B}$.

We define the set of all sequences from $\mathcal{B}$ as $\mathsf{Seq}^{\mathcal{B}}$:

$$\mathsf{Seq}^{\mathcal{B}} = \{\Gamma \mid \Gamma \text{ is a sequence from } \mathcal{B}\}.$$

3.2 Action

Next, we define four actions appearing in public-PEZ protocols. Assume that we have a sequence $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_m)$.

Suppose that players want to check the flavors of some candies $\frac{?}{c}$. In this case, the players must bite them to determine their flavor, and the eaten candies will disappear. We denote this action by (bite, T) for a set $T \subseteq \{1, 2, \ldots, m\}$ such that $\alpha_i = \frac{?}{0}$ or $\frac{?}{1}$ for every $i \in T$, where every candy in T is eaten and disappears. That is, the resulting sequence becomes $(\beta_1, \beta_2, \ldots, \beta_m)$ such that

$$\beta_i = \begin{cases} \frac{\mathsf{atom}(\alpha_i)}{\epsilon} & \text{if } i \in T \\ \alpha_i & \text{otherwise} \end{cases}$$

for every i, $1 \leq i \leq m$. For instance, for a sequence $(\frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{1}, \frac{?}{0})$, applying (bite, T) with a set $T = \{1, 3, 5\}$ results in $(\frac{1}{\epsilon}, \frac{?}{0}, \frac{1}{\epsilon}, \frac{?}{1}, \frac{0}{\epsilon})$.

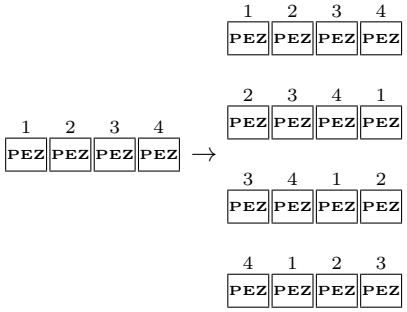
Suppose that players want to rearrange the order of sequence Γ. We denote this action by $(\mathsf{permute}, \pi)$ for a permutation $\pi \in S_m$, where S_m is the symmetric group of degree m. That is, the resulting sequence becomes $(\alpha_{\pi^{-1}(1)}, \alpha_{\pi^{-1}(2)}, \ldots, \alpha_{\pi^{-1}(m)})$.

Suppose that players want to perform a shuffling action such as shuffling the two dispensers, seen in Section 2. We denote this type of action by $(\mathsf{shuffle}, \Pi, \mathcal{F})$ for a subset $\Pi \subseteq S_m$ and a probability distribution $\mathcal{F}$ on Π. That is, defining $\mathsf{fixed}(\Pi) = \{j \mid 1 \leq j \leq m, \forall \sigma \in \Pi\ \sigma(j) = j\}$, the resulting sequence becomes $(\beta_1, \beta_2, \ldots, \beta_m)$ such that

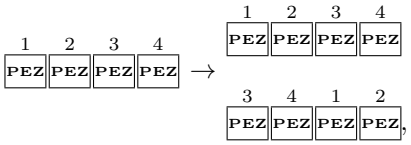
$$\beta_i = \begin{cases} \alpha_i & \text{if } i \in \mathsf{fixed}(\Pi) \\ \frac{?}{\mathsf{atom}(\alpha_{\pi^{-1}(i)})} & \text{if } i \notin \mathsf{fixed}(\Pi) \end{cases}$$

for every i, $1 \leq i \leq m$, where π is drawn from Π according to $\mathcal{F}$. When $\mathcal{F}$ is uniform, we write $(\mathsf{shuffle}, \Pi)$ by omitting it. Note that when an action $(\mathsf{shuffle}, \Pi, \mathcal{F})$ is applied to a sequence $(\alpha_1, \alpha_2, \ldots, \alpha_m)$, it should hold that $i \in \mathsf{fixed}(\Pi)$ for every i such that $\alpha_i = \frac{c}{\epsilon}$ for some c.

Herein, we introduce two shuffles that are also often used in card-based protocols. The first one is a "random cut" that shuffles a sequence cyclically. For instance, if a random cut is applied to a sequence of four candies, one of the following sequences is obtained. The probability of each occurrence is 1/4.



This shuffle is formally expressed as $(\mathsf{shuffle}, \{\mathsf{id}, (1\,2\,3\,4), (1\,2\,3\,4)^2, (1\,2\,3\,4)^3\})$ where id is the identity permutation. The second one is a "random bisection cut," which was implicitly seen in Section 2. Here, a sequence of candies is divided in half and the two sub-sequences are shuffled. For instance, applying a random bisection cut to a sequence of four candies results in



where each result occurs with a probability of 1/2. This shuffle is formally expressed as $(\mathsf{shuffle}, \{\mathsf{id}, (1\,3)(2\,4)\})$.

Finally, we define an action used at the end of a protocol. We use (result, p) for $p \in \{1, 2, \cdots, m\}$ to represent that the protocol is terminated and its output is the p-th candy.

3.3 Computational Model of Public-PEZ Protocols

In this subsection, we define a computational model of public-PEZ protocols via an abstract machine.

Let $\mathcal{B}$ be a box. Depending on the players' input, an initial sequence Γ_0 (from $\mathcal{B}$) is determined. By $U \subseteq \mathsf{Seq}^{\mathcal{B}}$, we denote the set of all possible input sequences.

Next, we define a visible sequence that represents all public information with regard to the flavors of the candies. Consequently, we define $\mathsf{top}(\frac{?}{c}) = ?$ and $\mathsf{top}(c) = \mathsf{top}(\frac{c}{\epsilon}) = c$ for $c \in \mathcal{B}$, which is based on the fact that when players bite a candy, the candy will disappear but its flavor will be memorized by all players. Then, we define $\mathsf{vis}(\Gamma)$ of a sequence $\Gamma = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ as

$$\mathsf{vis}((\alpha_1, \alpha_2, \ldots, \alpha_m)) = (\mathsf{top}(\alpha_1), \mathsf{top}(\alpha_2), \ldots, \mathsf{top}(\alpha_m)).$$

For instance,

$$\mathsf{vis}\left(\left(\frac{1}{\epsilon}, 0, \frac{?}{1}, \frac{0}{\epsilon}, \frac{1}{\epsilon}, \frac{?}{0}, 1\right)\right) = (1, 0, ?, 0, 1, ?, 1).$$

Furthermore, we define the set $\mathsf{Vis}^{\mathcal{B}}$ of all visible sequences as

$$\mathsf{Vis}^{\mathcal{B}} = \{\mathsf{vis}(\Gamma) \mid \Gamma \in \mathsf{Seq}^{\mathcal{B}}\}.$$

We are now ready to formally define a public-PEZ protocol. A *protocol* is a 4-tuple $\mathcal{P} = (\mathcal{B}, U, A, Q)$ such that

- $\mathcal{B}$ is a box;
- $U \subseteq \mathsf{Seq}^{\mathcal{B}}$ is a set of input sequences;
- Q is a set of states, containing the initial state q_0 and final state q_f;

– $A : (Q - \{q_f\}) \times \mathsf{Vis}^{\mathcal{B}} \to Q \times \texttt{Action}$ is an action function, where `Action` is the set of all possible actions (bite, T), $(\mathsf{permute}, \pi)$, $(\mathsf{shuffle}, \Pi, \mathcal{F})$, and (result, p).

A protocol $\mathcal{P} = (\mathcal{B}, U, A, Q)$ proceeds as imagined; starting with an initial sequence $\Gamma_0 \in U$ and initial state q_0, it changes the sequence and state based on the output of the action function. When the state becomes q_f, the protocol $\mathcal{P}$ terminates with an action (result, p) for some p.

4 Formal Description of Our AND Protocol and Another One

In this section, we formally describe our AND protocol presented in Section 2 based on the computational model of public-PEZ protocols defined in Section 3, which can be described as follows.

The five-candy AND protocol

Box:

$$\mathcal{B} = [\, 3 \cdot 0, 2 \cdot 1 \,]$$

Input:

$$U = \left\{ \left(\frac{?}{0}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}\right), \left(\frac{?}{0}, \frac{?}{0}, \frac{?}{1}, \frac{?}{1}, \frac{?}{0}\right), \left(\frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{0}, \frac{?}{1}\right), \left(\frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}\right) \right\}$$

Steps:

1. $(\mathsf{permute}, (1\,3))$
2. $(\mathsf{shuffle}, \{\mathsf{id}, (2\,3)(4\,5)\})$
3. $(\mathsf{bite}, \{4\})$
4. if visible seq.$= (?, ?, ?, 0, ?)$ then $(\mathsf{result}, 3)$
5. $\mathsf{else\,if}$ visible seq.$= (?, ?, ?, 1, ?)$ then $(\mathsf{result}, 2)$

Next, to display another formal protocol, we describe the XOR protocol based on the card-based XOR protocol [22].

The XOR protocol based on [22]

Box:

$$\mathcal{B} = [\, 7 \cdot 0, 7 \cdot 1 \,]$$

Input:

$$U = \left\{ \begin{array}{l} \left(\frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, 0, 0, 1, 1\right), \\ \left(\frac{?}{0}, \frac{?}{1}, \frac{?}{1}, \frac{?}{0}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, 0, 0, 1, 1\right), \\ \left(\frac{?}{1}, \frac{?}{0}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, 0, 0, 1, 1\right), \\ \left(\frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, \frac{?}{0}, \frac{?}{1}, 0, 0, 1, 1\right) \end{array} \right\}$$

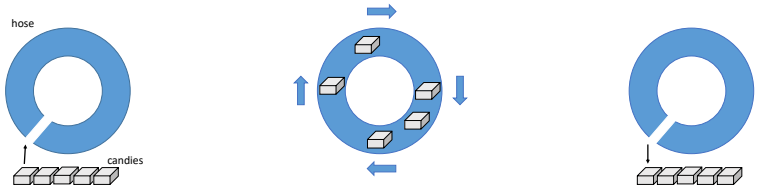
Steps:

1. $(\mathsf{shuffle}, \{\mathsf{id}, (7\,8\,9\,10), (7\,8\,9\,10)^2, (7\,8\,9\,10)^3\})$
2. $(\mathsf{shuffle}, \{\mathsf{id}, (1\,2\,3\,4), (1\,2\,3\,4)^2, (1\,2\,3\,4)^3\})$
3. $(\mathsf{shuffle}, \{\mathsf{id}, (5\,6\,7\,8), (5\,6\,7\,8)^2, (5\,6\,7\,8)^3\})$
4. $(\mathsf{shuffle}, \{\mathsf{id}, (1\,5)\})$
5. $(\mathsf{bite}, \{1, 5\})$
6. if visible seq.$= (0, ?, ?, ?, 0, ?, ?, ?, \cdots)$ then $(\mathsf{permute}, (1\,11)(5\,12))$
   go to step 2
   else if visible seq.$= (1, ?, ?, ?, 1, ?, ?, ?, \cdots)$ then $(\mathsf{permute}, (1\,13)(5\,14))$
   go to step 2
   else if visible seq.$= (0, ?, ?, ?, 1, ?, ?, ?, \cdots)$ or $(1, ?, ?, ?, 0, ?, ?, ?, \cdots)$
   go to step 7
7. $(\mathsf{bite}, \{3, 7\})$
8. if visible seq.$= (c, ?, 0, ?, \overline{c}, ?, 1, ?, \cdots)$ or $(c, ?, 1, ?, \overline{c}, ?, 0, ?, \cdots)$
   for some $c \in \{0, 1\}$ then $(\mathsf{result}, 9)$
   else if visible seq.$= (c, ?, 0, ?, \overline{c}, ?, 0, ?, \cdots)$ or $(c, ?, 1, ?, \overline{c}, ?, 1, ?, \cdots)$
   for some $c \in \{0, 1\}$ then $(\mathsf{result}, 10)$

We describe the above XOR protocol as an example. However, we do not intend to use this practically because it is relatively complicated, has a loop, and is a Las Vegas algorithm. (Actually, we can obtain a simple finite-runtime XOR protocol based on the four-card XOR protocol in [21].) Note that the number of available candies is finite, *i.e.*, the box $\mathcal{B}$ has exactly seven lemon candies and seven orange candies. Therefore, if there is no candy available in Step 6, the protocol fails. This is a big difference between card-based protocols and public-PEZ protocols; the Las Vegas algorithm does not work well in public-PEZ cryptography.

5 Implementations of Shuffles of Candies

In this section, we discuss the feasibility of shuffling actions in public-PEZ protocols. It is more difficult to shuffle a sequence of candies using the players'



(a) Put candies in a hose. (b) Rotate the hose. (c) Take out the candies.

Fig. 7: How to implement a random cut.

hands, compared to shuffling a sequence of cards. Therefore, we consider using the following special tools to implement a random bisection cut and random cut on a sequence of candies.

Random bisection cut. As already seen in Section 2, we use two identical dispensers. First, the two divided sequences of candies are packed into the two dispensers without changing the orders. Then, we shuffle the two dispensers, take out the candies from each of the dispensers, and arrange them.
The operation of shuffling the two dispensers must be implemented so that nobody knows how many times they are switched. The previous research [32] proposed a secure implementation method for a random bisection cut in card-based protocols using a Styrofoam ball. This method can be used in public-PEZ protocols as well. Specifically, two dispensers are placed in a Styrofoam ball (the contents of which cannot be seen from the outside). Then, the ball is thrown up to shuffle the dispensers inside the ball.

Random cut. We consider using a hose as shown in Figure 7. First, we place the candies (to be shuffled) in the hose while maintaining the order, and tape its inlet and outlet. Then, we move the candies by rotating the hose. Finally, we take out the candies from the hose, while maintaining the order, and arrange them. The diameter of the hose must be tight enough that the order of the candies inside the hose does not change when it is rotated.

6 Conclusion

In this study, we designed public-PEZ cryptography. We constructed its computational model and presented a few protocols within the model. Further, we discussed the feasibility of shuffling actions for a sequence of PEZ candies. As public-PEZ protocols do not require private actions, players can perform MPC more securely and easily. In particular, we believe that our AND protocol presented in Section 2 is practical enough to be utilized in daily activities.

People might assume that this paper would just replace "cards" in card-based cryptography with "candies and dispensers," *i.e.*, public-PEZ cryptography is a type of re-implementation of card-based cryptography, and thus, there would

not be much novelty. However, we believe that this is not the case. As already demonstrated, unlike a deck of cards, one cannot turn a candy face down and a candy disappears after one confirms its value; therefore, we need novel and careful treatment to construct a rigorous model. In addition, public-PEZ cryptography is simple, which is a virtue.

Acknowledgement

We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported in part by JSPS KAKENHI Grant Number JP19J21153.

References

1. Abe, Y., Iwamoto, M., Ohta, K.: Efficient private PEZ protocols for symmetric functions. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. LNCS, vol. 11891, pp. 372–392. Springer, Cham (2019), 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), 69–84 (2003), https://doi.org/10.1016/S0304-3975(03)00210-X
3. Bultel, X., Dreier, J., Dumas, J., Lafourcade, P., Miyahara, D., Mizuki, T., Nagao, A., Sasaki, T., Shinagawa, K., Sone, H.: Physical zero-knowledge proof for Makaro. In: Stabilization, Safety, and Security of Distributed Systems. LNCS, vol. 11201, pp. 111–125 (2018), https://doi.org/10.1007/978-3-030-03232-6_8
4. D'Arco, P., De Prisco, R.: Secure computation without computers. Theor. Comput. Sci. **651**, 11–36 (2016), https://doi.org/10.1016/j.tcs.2016.08.003
5. Den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology—EUROCRYPT '89. LNCS, vol. 434, pp. 208–217. Springer, Berlin, Heidelberg (1990), https://doi.org/10.1007/3-540-46885-4_23
6. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: Du, D.Z., Duan, Z., Tian, C. (eds.) Computing and Combinatorics. LNCS, vol. 11653, pp. 166–177. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-26176-4_14
7. Francis, D., Aljunid, S.R., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Necessary and sufficient numbers of cards for securely computing two-bit output functions. In: Phan, R.C.W., Yung, M. (eds.) Paradigms in Cryptology–Mycrypt 2016. Malicious and Exploratory Cryptology. LNCS, vol. 10311, pp. 193–211. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-61273-7_10
8. Heather, J., Schneider, S., Teague, V.: Cryptographic protocols with everyday objects. Formal Aspects Comput. **26**(1), 37–62 (2014), https://doi.org/10.1007/s00165-013-0274-7
9. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology—ASIACRYPT 2017. LNCS, vol. 10626, pp. 126–155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5

10. Koch, A., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology–ASIACRYPT 2019. LNCS, vol. 11921, pp. 488–517. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-34578-5_18
11. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology—ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-662-48797-6_32
12. Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: A physical ZKP for Slitherlink: How to perform physical topology-preserving computation. In: Heng, S.H., Lopez, J. (eds.) Information Security Practice and Experience. LNCS, vol. 11879, pp. 135–151. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-34339-2_8
13. Miyahara, D., ichi Hayashi, Y., Mizuki, T., Sone, H.: Practical card-based implementations of Yao's millionaire protocol. Theor. Comput. Sci. **803**, 207–221 (2020), https://doi.org/10.1016/j.tcs.2019.11.005
14. Miyahara, D., Hayashi, Y.i., Mizuki, T., Sone, H.: Practical and easy-to-understand card-based implementation of yao's millionaire protocol. In: Kim, D., Uma, R.N., Zelikovsky, A. (eds.) Combinatorial Optimization and Applications. LNCS, vol. 11346, pp. 246–261. Springer, Cham (2018), https://doi.org/10.1007/978-3-030-04651-4_17
15. Miyahara, D., Robert, L., Lafourcade, P., Takeshige, S., Mizuki, T., Shinagawa, K., Nagao, A., Sone, H.: Card-based ZKP protocols for Takuzu and Juosan. In: Farach-Colton, M., Prencipe, G., Uehara, R. (eds.) Fun with Algorithms. Leibniz International Proceedings in Informatics (LIPIcs), vol. 157, pp. 20:1–20:21. Schloss Dagstuhl, Dagstuhl, Germany (2020), https://doi.org/10.4230/LIPIcs.FUN.2021.20
16. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theor. Comput. Sci. **622**(C), 34–44 (2016), https://doi.org/10.1016/j.tcs.2016.01.039
17. Mizuki, T., Kugimoto, Y., Sone, H.: Secure multiparty computations using a dial lock. In: Cai, J.Y., Cooper, S.B., Zhu, H. (eds.) Theory and Applications of Models of Computation. LNCS, vol. 4484, pp. 499–510. Springer, Berlin, Heidelberg (2007), https://doi.org/10.1007/978-3-540-72504-6_45
18. Mizuki, T., Kugimoto, Y., Sone, H.: Secure multiparty computations using the 15 puzzle. In: Dress, A., Xu, Y., Zhu, B. (eds.) Combinatorial Optimization and Applications. LNCS, vol. 4616, pp. 255–266. Springer, Berlin, Heidelberg (2007), https://doi.org/10.1007/978-3-540-73556-4_28
19. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. Int. J. Inf. Secur. **13**(1), 15–23 (2014), https://doi.org/10.1007/s10207-013-0219-4
20. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E100.A**(1), 3–11 (2017), https://doi.org/10.1587/transfun.E100.A.3
21. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics. LNCS, vol. 5598, pp. 358–369. Springer, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-642-02270-8_36
22. Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. The Australasian Journal of Combinatorics **36**, 279–293 (2006), https://ajc.maths.uq.edu.au/?page=get_volumes&volume=36

23. 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
24. Nakai, T., Shirouchi, S., Iwamoto, M., Ohta, K.: Four cards are sufficient for a card-based three-input voting protocol utilizing private permutations. In: Shikata, J. (ed.) Information Theoretic Security. LNCS, vol. 10681, pp. 153–165. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-72089-0_9
25. Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires' problem utilizing private permutations. In: Foresti, S., Persiano, G. (eds.) Cryptology and Network Security. LNCS, vol. 10052, pp. 500–517. Springer, Cham (2016), https://doi.org/10.1007/978-3-319-48965-0_30
26. Ono, H., Manabe, Y.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: Asia Joint Conference on Information Security (AsiaJCIS). pp. 23–28 (2018), https://doi.org/10.1109/AsiaJCIS.2018.00013
27. Ono, H., Manabe, Y.: Card-based cryptographic protocols with the minimum number of rounds using private operations. In: Pérez-Solà, C., Navarro-Arribas, G., Biryukov, A., Garcia-Alfaro, J. (eds.) Data Privacy Management, Cryptocurrencies and Blockchain Technology. LNCS, vol. 11737, pp. 156–173. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-31500-9_10
28. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Numberlink. In: Farach-Colton, M., Prencipe, G., Uehara, R. (eds.) Fun with Algorithms. Leibniz International Proceedings in Informatics (LIPIcs), vol. 157, pp. 22:1–22:11. Schloss Dagstuhl, Dagstuhl, Germany (2020), https://doi.org/10.4230/LIPIcs.FUN.2021.22
29. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for Sudoku. Theor. Comput. Sci. **839**, 135–142 (2020), https://doi.org/10.1016/j.tcs.2020.05.036
30. Takashima, K., Abe, Y., Sasaki, T., Miyahara, D., Shinagawa, K., Mizuki, T., Sone, H.: Card-based secure ranking computations. In: Li, Y., Cardei, M., Huang, Y. (eds.) Combinatorial Optimization and Applications. LNCS, vol. 11949, pp. 461–472. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-36412-0_37
31. Takashima, K., Miyahara, D., Mizuki, T., Sone, H.: Card-based protocol against actively revealing card attack. In: Martín-Vide, C., Pond, G., Vega-Rodríguez, M.A. (eds.) Theory and Practice of Natural Computing. LNCS, vol. 11934, pp. 95–106. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-34500-6_6
32. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. Int. J. Inf. Secur. **19**(4), 445–452 (2020), https://doi.org/10.1007/s10207-019-00463-w
33. Yasunaga, K.: Practical card-based protocol for three-input majority. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E103.A**(11), 1296–1298 (2020), https://doi.org/10.1587/transfun.2020EAL2025