



A Physical Zero-Knowledge Proof for Sumplete, a Puzzle Generated by ChatGPT

Kyosuke Hatsugai^{1(✉)}, Kyoichi Asano¹, and Yoshiki Abe^{1,2}

¹ The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu,
Tokyo 182-8585, Japan

{hatsugai,k.asano,yoshiki}@uec.ac.jp

² National Institute of Advanced Industrial Science and Technology, 2-3-26 Aomi,
Koto-ku, Tokyo 135-0064, Japan

Abstract. In March 2023, ChatGPT generated a new puzzle, Sumplete. Sumplete consists of an $n \times n$ grid, each whose cell has an integer. In addition, each row and column of the grid has an integer, which we call a *target value*. The goal of Sumplete is to make the sum of integers in each row and column equal to the target value by deleting some integers of the cells. In this paper, we prove that Sumplete is NP-complete and propose a physical zero-knowledge proof for Sumplete. To show the NP-completeness, we give a polynomial reduction from the subset sum problem to Sumplete. In our physical zero-knowledge proof protocol, we use a card protocol that realizes the addition of negative and positive integers using cyclic permutation on a sequence of cards. To keep the solution secret, we use a technique named *decoy technique*.

Keywords: Physical Zero-knowledge Proof · Card-based Cryptographic Protocol · Sumplete

1 Introduction

1.1 Background

Chat Generative Pre-trained Transformer (ChatGPT) is an artificial intelligence chatbot developed by OpenAI [18]. ChatGPT can respond to questions more naturally than usual artificial intelligence. Moreover, it can work on generative tasks such as writing sentences, programming, drawing pictures and making puzzles.

In March 2023, ChatGPT generated a puzzle named *Sumplete*¹ [19]. Sumplete is a puzzle consisting of a grid with $n \times n$ cells. Each cell in the grid has an integer. In addition, each row and column in the grid also has an integer, and we call it the *target value* hereafter. The goal of Sumplete is to delete some

¹ This name was also named by ChatGPT [19].

-3	3	2	-7	9	-8
5	-1	-9	8	-7	-5
1	8	6	3	5	14
9	9	-6	8	-8	3
-7	3	-9	5	8	16
11	10	-7	6	0	

-3	✗	2	-7	✗	-8
5	-1	-9	✗	✗	-5
✗	8	6	✗	✗	14
9	✗	-6	8	-8	3
✗	3	✗	5	8	16
11	10	-7	6	0	

Fig. 1. Example of a problem (left) and its solution (right) of Sumplete.

integers in the cells so that the sum of the non-deleted integers in every row (resp., column) is equal to the target value of the row (resp., column). Figure 1 shows an example of the Sumplete problem and its solution: deleted integers' locations.

Since there are 2^{n^2} possible combinations of locations to delete integers, it seems hard to judge whether the given Sumplete problem has a solution if n becomes large. Indeed, as described later, Sumplete is NP-complete. It is interesting that AI generates an NP-complete puzzle.

This paper considers a situation where a contestant wants to convince a challenger the existence of a solution to a given Sumplete problem while the contestant does not want to tell the solution to the challenger. Although these requirements may seem contradictory at first glance, they can be satisfied simultaneously using a cryptographic technique called Zero-knowledge proof (ZKP).

ZKP is an interactive proof proposed by Goldwasser, Micali, and Rackoff in 1989 [7]. ZKP allows a prover P , who knows the solution to a problem, to convince a verifier V of the existence of the problem's solution without revealing the solution itself. Usually, ZKP protocols are implemented using computers. However, ZKP protocols implemented by physical tools like cards instead of computers are also studied, called *physical zero-knowledge proof* (physical ZKP). The first physical ZKP protocol is that for a pencil puzzle, Sudoku, proposed by Gradwohl, Naor, Pinkas, and Rothblum in 2007 [8]. Since then, physical ZKP protocols are proposed for various puzzles such as Sudoku [8, 20, 28, 29], Nonogram [4, 21], Slitherlink [11, 12], Akari [2], Numberlink [23, 24], Norinori [6], Makaro [3, 27], Takuzu [2, 14], Kakuro [2, 15], Shikaku [26], and Pancake sorting [10].

Most card operations used in physical ZKP protocols come from card-based cryptography, secure multiparty computation (MPC) using cards. The study of card-based cryptographic protocol began with the protocol for computing the two-input logical AND function by den Boar in 1989 [1] and that for computing the two-input logical XOR function by Crepeau and Kilian in 1993 [5]. After their work, to realize MPC with cards for more complex functions, card shuffle operations called pile-shifting shuffle [30, 31] and pile-scramble shuffle [9] are proposed. We also use these operations in this paper.

1.2 Our Contributions

Neither NP-completeness of Sumplete nor zero-knowledge proof protocol for Sumplete has been proven. In this paper, we prove that Sumplete is NP-complete. Our proof is based on the reduction from the Subset Sum Problem (SSP), known as an NP-complete problem.

In addition, we propose a physical zero-knowledge proof protocol for Sumplete. Our card-based ZKP protocol allows a prover P to convince a verifier V that integers in a Sumplete instance's cells can be deleted to satisfy the sum conditions in each row and column. On the other hand, during the protocol, the verifier V cannot get any information about which integers in the cells are deleted. Because of the NP-completeness of Sumplete, the larger problem's size n becomes, the more difficult it is to obtain the solution. Therefore, it is worth proposing a zero-knowledge proof protocol for Sumplete.

1.3 Organization

The rest of this paper is organized as follows. In Sect. 2, we introduce zero-knowledge proof and card operations. Section 3 is devoted to show the NP-completeness of Sumplete. In Sect. 4, we propose a physical ZKP protocol for Sumplete. Then, in Sect. 5, we show our proposed protocol satisfies the conditions required for ZKP. Finally, Sect. 6 concludes this paper.

2 Preliminaries

2.1 Notation

(Multi)sets are denoted by uppercase letters of the italic font, e.g., $\mathcal{A} = \{a_1, a_2, a_3\}$. Vectors are denoted by lowercase letters of the bold font and the i -th element of a vector \mathbf{v} is denoted by v_i , e.g., $\mathbf{v} = (v_1, v_2, v_3)$. Matrices are denoted by uppercase letters of the bold font and the element of the i -th row and j -th column of a matrix \mathbf{M} is denoted by $m_{i,j}$, e.g.,

$$\mathbf{M} = \begin{pmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{pmatrix}.$$

2.2 Zero-Knowledge Proof








Zero-knowledge proof is an interactive proof between a prover P and a verifier V . From now, we assume that P is a probabilistic Turing machine with unbounded computational ability and V is a probabilistic polynomial-time Turing machine. Let x be an instance of a NP language. For a given x which has the witness, we suppose that P can calculate the witness from x , however V cannot. We note that x has the witness if and only if P knows the witness since the computational ability of P is unbound. In zero-knowledge proof, P interacts with V and finally convince V that the problem x has the witness without revealing the witness. Zero-knowledge proof protocols must achieve the following three conditions.

Completeness. If P knows the witness, V is always convinced.

Soundness. If P does not know the witness, V is not convinced with more than negligible probability. The probability that V is convinced when P does not know the witness is called *soundness error*. If soundness error is less than 1, it approaches asymptotically to 0 by executing proof many times. However, repeating physical protocols by human hands is hard. Therefore, it is desirable that soundness error of physical zero-knowledge proof is 0.

Zero Knowledge. V cannot obtain any information of the witness. Formally, for every V , there exists a probabilistic polynomial-time algorithm S that does not know the witness. If the output of S is indistinguishable from the output of the interaction of P and V , any information about the solution is not leaked during the interaction.

2.3 Card Operations

In this paper, we use cards whose front side is either  or  and whose back side is . We assume that the front sides of cards with the same suit are identical, i.e., we cannot distinguish them. In addition, the back sides of all cards are also assumed to be indistinguishable. For understanding, we denote a face-down card  (resp., ) by  (resp., ) with a small suit symbol below the question mark.

Cyclic Shift. Let a $\mathbf{c} := (c_1, c_2, \dots, c_k)$ be a card sequence of k cards. *Left cyclic shift* over \mathbf{c} is defined as a operation that outputs

$$(c_{\rho^{-1}(1)}, c_{\rho^{-1}(2)}, \dots, c_{\rho^{-1}(k)}),$$

where ρ is a cyclic permutation $\rho := (1 \ k \ k - 1 \ \dots \ 3 \ 2)$.

Similarly, *right cyclic shift* over \mathbf{c} is defined as a operation that outputs

$$(c_{\sigma^{-1}(1)}, c_{\sigma^{-1}(2)}, \dots, c_{\sigma^{-1}(k)}),$$

where σ is a cyclic permutation $\sigma := (1 \ 2 \ 3 \ \dots \ k - 1 \ k)$.

Pile-Scramble Shuffle. Let a $\mathbf{p} := (p_1, p_2, \dots, p_k)$ be a sequence of k piles of cards. Note that each pile has the same number of cards. *Pile-scramble shuffle* over \mathbf{p} is a operation that outputs

$$(p_{\pi^{-1}(1)}, p_{\pi^{-1}(2)}, \dots, p_{\pi^{-1}(k)}),$$

where a permutation π is an element of S_k , the symmetric group of degree k .

2.4 Representation of an Integer

Here, we show a representation of an integer using cards. In physical ZKP protocols and card-based cryptography protocols, e.g., [10, 13, 22, 25, 32], integers from

1 to n are represented by a sequence of n cards, which consists of a \heartsuit card and $n - 1$ \clubsuit cards. Specifically, an integer i ($1 \leq i \leq n$) is represented by the position of \heartsuit in the sequence: if \heartsuit is at the i -th leftmost position, the sequence represents the integer i . For example, 1 and 3 are represented as follows.

$$\begin{aligned}
 1 &= \heartsuit \clubsuit \clubsuit \clubsuit \clubsuit \\
 3 &= \clubsuit \clubsuit \heartsuit \clubsuit \clubsuit
 \end{aligned}$$

In this paper, we apply this method to represent integers including less than 1.

Definition 1 (Integer Counter). Let α and β be positive integers. An integer i ($-\alpha \leq i \leq \beta$) is represented by a sequence of $\alpha + \beta + 1$ cards, consisting of a \heartsuit card and $\alpha + \beta$ \clubsuit cards. Specifically, if \heartsuit is at the i -th leftmost position of the sequence, the sequence represents the integer $i - \alpha - 1$. We call the card sequence to represent an integer an *integer counter*.

For example, 0 and -2 are represented as follows when $\alpha = 3$ and $\beta = 2$.

$$\begin{aligned}
 0 &= \clubsuit \clubsuit \clubsuit \heartsuit \clubsuit \clubsuit \\
 -2 &= \clubsuit \heartsuit \clubsuit \clubsuit \clubsuit \clubsuit
 \end{aligned}$$

We execute the addition by shifting the counter. This method is used by Shinagawa et al. [31] and Ruangwises and Itoh [26]. Suppose there is an integer counter representing $x \in \mathbb{Z}$, where the number of cards (i.e., the values α and β for the counter) is large enough so that \heartsuit does not overflow. Then, we can obtain the counter representing $x + y$ ($y \in \mathbb{Z}$) as follows: if $y < 0$, we execute the *left* cyclic shift over the card sequence $|y|$ times; otherwise (i.e., if $y \geq 0$), we execute the *right* cyclic shift over the card sequence $|y|$ times. Since these cyclic shift operations can be performed even if the cards of the counter are face-down, the addition of y can be performed without revealing the value of x .

3 NP-Completeness of Sumplete

In this section, we prove that Sumplete is NP-complete. To prove the NP-completeness, we show a polynomial-time reduction from an NP-complete problem SSP to Sumplete.

3.1 Formal Definition of Problems

Before proving Sumplete’s NP-completeness, we formally define the decisional version of Sumplete and SSP.

Definition 2 (Sumplete). An instance of Sumplete consists of three ingredients: an $n \times n$ matrix $\mathbf{G} \in \mathbb{Z}^{n \times n}$ that represents each integer of the corresponding cell, a vector $\mathbf{R} \in \mathbb{Z}^n$ that represents each row’s target value, and a vector

$\mathbf{C} \in \mathbb{Z}^n$ that represents each column's target value. The answer of the instance $S = (\mathbf{G}, \mathbf{R}, \mathbf{C})$ is *Yes* if there exists an $n \times n$ matrix $\hat{\mathbf{G}} \in \{0, 1\}^{n \times n}$ that satisfies following equations:

$$r_i = \sum_{j=1}^n g_{i,j} \hat{g}_{i,j} \quad \text{for all } i \in \{1, \dots, n\},$$

$$c_j = \sum_{i=1}^n g_{i,j} \hat{g}_{i,j} \quad \text{for all } j \in \{1, \dots, n\}.$$

If there does not exist such $\hat{\mathbf{G}}$, the answer is *No*.

For instance, the example of Fig. 2 can be represented as follows:

$$\mathbf{G} = \begin{pmatrix} -3 & 3 & 2 & -7 & 9 \\ 5 & -1 & -9 & 8 & -7 \\ 1 & 8 & 6 & 3 & 5 \\ 9 & 9 & -6 & 8 & -8 \\ -7 & 3 & -9 & 5 & 8 \end{pmatrix},$$

$$\mathbf{R} = (-8 \ -5 \ 14 \ 3 \ 16),$$

$$\mathbf{C} = (11 \ 10 \ -7 \ 6 \ 0).$$

The answer of above instance is *Yes* since there exists the following solution $\hat{\mathbf{G}}$:

$$\hat{\mathbf{G}} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Definition 3 (Subset Sum Problem). The Subset Sum Problem (SSP) consists of a multiset $\mathcal{A} \subset \mathbb{Z}^n$ and an integer $N \in \mathbb{Z}$. The answer of the instance $SSP = (\mathcal{A}, N)$ is *Yes* if there exists a subset $\mathcal{A}' \subseteq \mathcal{A}$ that satisfies $\sum_{a \in \mathcal{A}'} a = N$. If there does not exist such \mathcal{A}' , the answer is *No*.

For example, let us consider the instance $SSP = (\mathcal{A}, N)$ where $\mathcal{A} := \{-3, 3, 2, -7, 9\}$ and $N := -8$. The answer of this instance is *Yes* since there exists the solution $\mathcal{A}' = \{-3, 2, -7\} \subset \mathcal{A}$.

3.2 Proof of NP-Completeness

To show that Sumplete is NP-complete, we prove that the following holds.

- (1) Sumplete is in NP.
- (2) Sumplete is polynomial-time reductive from SSP.

Proof of (1). We prove the existence of a non-deterministic polynomial-time algorithm which can decide whether Yes or No for a given instance of Sumplete. Let us consider the non-deterministic algorithm M that works as follows.

1. M non-deterministically chooses some cells.
2. M deletes integers in chosen cells.
3. For every row and column, M calculates the sum of non-deleted integers and compares it with the target value. M rejects if there are rows or columns whose sum does not equal the target value. Otherwise, M accepts.

Since each operation ends in polynomial time, M halts in polynomial time. Thus, (1) holds.

Proof of (2). We prove (2) by showing the following three conditions hold.

- (i) There exists a polynomial-time reduction f from SSP to Sumplete.
- (ii) For arbitrary SSP's instance SSP , if the answer of SSP is Yes, then the answer of the reduced Sumplete's instance $S':=f(SSP)$ is Yes.
- (iii) For arbitrary SSP's instance SSP , if the answer of $S':=f(SSP)$ is Yes, the answer of SSP is Yes.

First, we show (i). Let us consider the following polynomial-time reduction f (See also Fig. 2).

- f receives an instance of SSP, denoted by $SSP:=(\mathcal{A}, N)$ where $\mathcal{A}:=\{a_1, a_2, \dots, a_n\} \in \mathbb{Z}^n$ and $N \in \mathbb{Z}$.
- f outputs an instance of Sumplete, denoted by $S':=(\mathbf{G}', \mathbf{R}', \mathbf{C}')$, where \mathbf{G}' , \mathbf{R}' , and \mathbf{C}' are defined as follows:

$$\mathbf{G}' := \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ a_{\rho^{-1}(1)} & a_{\rho^{-1}(2)} & \cdots & a_{\rho^{-1}(n)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{\rho^{-(n-1)}(1)} & a_{\rho^{-(n-1)}(2)} & \cdots & a_{\rho^{-(n-1)}(n)} \end{pmatrix},$$

$$\mathbf{R}' := (N \ N \ \cdots \ N),$$

$$\mathbf{C}' := (N \ N \ \cdots \ N),$$

where $\rho := (1 \ n \ n-1 \ \cdots \ 3 \ 2)$ is a cyclic permutation.

We note that all the target values in S' are N . In addition, each element of \mathcal{A} appears once for each row and column. In reduction f , $n^2 + 2n$ times writing of integers and n times shifting are executed. Therefore, the running time of f is polynomial in n .

Next, we show (ii). Since the answer of SSP is Yes, there exists the solution $\mathcal{A}' \subseteq \mathcal{A}$ such that $\sum_{a' \in \mathcal{A}'} a' = N$. Here, let us consider to delete all integers in \mathbf{G}' excluding all integers in \mathcal{A}' . Then, the set of the non-deleted integers for each row and column in \mathbf{G}' equals \mathcal{A}' since each row and column in \mathbf{G}' equals

a_1	a_2	\cdots	a_n	N
a_2	a_3	\cdots	a_1	N
\vdots	\vdots	\ddots	\vdots	\vdots
a_n	a_1	\cdots	a_{n-1}	N
N	N	\cdots	N	

Fig. 2. Instance of Sumplete constructed from an instance of the subset sum problem.

A. Thus, the sum of the non-deleted integers for each row and column is N . Therefore, the answer of S' is Yes.

Finally, we show (iii). For each row and column in S' after deletion, the set of non-deleted integers equals the solution of SSP. Therefore, if the answer of S' is Yes, then the answer of SSP is Yes.

From (i), (ii), and (iii), (2) holds. □

Hence, from (1) and (2), Sumplete is NP-complete.

4 Physical Zero-Knowledge Proof Protocol for Sumplete

In our proposed protocol, the prover represents the solution of a Sumplete’s instance using cards, and the verifier checks whether the sum of non-deleted integers equals the target value for each row and column.

4.1 Idea of Proposed Protocol

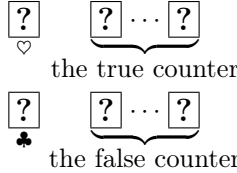
The outline of our proposed protocol is as follows. The prover calculates the sum of non-deleted integers for each row and column of the grid using an integer counter. Then, the verifier checks that the sum equals the target value. If the sum equals the target value for all rows and columns, the verifier can be convinced of the solution’s existence.

Decoy Technique. To realize the above operation without revealing the solution, i.e., the locations of cells whose integer is deleted, we prepare two integer counters called a *true counter* and a *false counter* for each row and column. The true (resp., false) counter is used to calculate the sum of the non-deleted (resp., deleted) integers in a row or column. In our protocol, for each integer in a row or column, the prover add the integer to the true or false counter depending on the solution: if it is a non-deleted (resp., deleted) integer, it is added to the true (resp., false) counter. By using a technique we call *decoy technique*, we can add integers while hiding the solution, i.e., which counter they were added to. Similar technique is widely seen in physical cryptography, e.g., [16, 17].

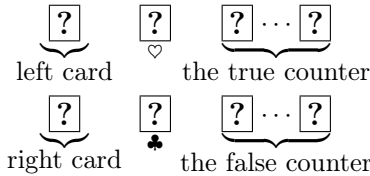
4.2 Proposed Protocol

Proposed protocol proceeds as follows.

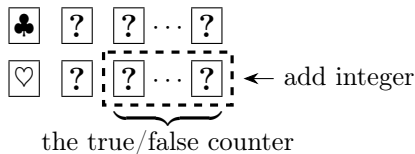
1. For each cell, the prover places a pair of face-down cards $\boxed{?} \boxed{?}$ on non-deleted integer's cells and $\boxed{?} \boxed{?}$ on deleted integer's cells.
2. For each row and column, the prover and the verifier execute the following operations.
 - (a) The prover calculates α (resp., β), an absolute value of the sum of negative (resp., positive) integers. The prover also makes a true counter and a false counter which can represent an integer i ($-\alpha \leq i \leq \beta$). Then, the prover places the false counter below the true counter. In addition, the verifier checks that both counters indicates 0.
 - (b) The prover places $\boxed{\heartsuit}$ on the left of the true counter and $\boxed{\clubsuit}$ on the left of the false counter. After placing them, the prover makes all the cards face-down. We call these two sequences of cards a *card matrix*.



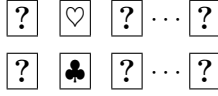
- (c) For every cell in the row (or column), the prover and the verifier execute the following operations.
 - i. The prover picks a pair of cards on the cell and places left (resp., right) card of the pair to leftmost position of the upper (resp., lower) sequence of the card matrix made in Step 2(b).



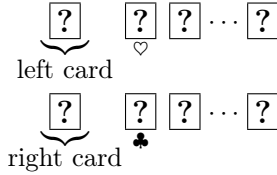
- ii. The prover regards each of the upper row and lower row in the matrix made in Step 2(c)i as a pile and applies pile-scramble shuffle to these two piles.
- iii. The prover opens the leftmost card of each row and adds the integer of the cell to the counter in the row whose opened leftmost card is $\boxed{\heartsuit}$. Note that this addition must be executed keeping the cards of the counter face-down.



- iv. The prover makes all cards face-down. The prover regards upper and lower row as two piles and applies pile-scramble shuffle in same way as Step 2(c)ii.
- v. The verifier opens the second leftmost card of each row. The verifier regards the upper and lower row as two piles and replaces these rows so that the row in which \heartsuit is opened becomes the upper row.



- vi. The prover makes all the cards face-down and returns the pair of leftmost cards of the rows to the cell.



- (d) The verifier opens the leftmost card of each row and opens the counter in the row whose opened leftmost card is \heartsuit card. If the value indicated by opened counter does not equal the target value, the verifier rejects.

3. If the verifier did not reject for all rows and columns, the verifier accepts.

4.3 Decoy Technique

The decoy technique is used in Step 2(c)ii and Step 2(c)iii. In Step 2(c)iii, the counter in the same row with \heartsuit is the true (resp., false) counter if the integers' cell is non-deleted (resp., deleted). Thanks to the pile-scramble shuffle in Step 2(c)ii, the location of \heartsuit in the leftmost column is independent from the card matrix made in Step (c)i. Therefore, the verifier cannot know which counter appears on the right of \heartsuit .

4.4 The Numbers of Cards and Shuffles

At first, we consider the number of cards used in our protocol. In Step 1, the prover uses n^2 pairs of \heartsuit and \clubsuit cards. In Step 2, the prover and verifier check n rows and n columns. Let M be the max value of $\alpha + \beta$ in each row and column, where α and β are those in Step 2(a). Then, Step 2(a) needs a \heartsuit card and at most M \clubsuit cards for each counter. In addition, Step 2(b) needs a \heartsuit card and a \clubsuit card. Since we can reuse the cards required for Steps 2(a) and 2(b), it is sufficient to consider the maximum number of cards required for the counter within the $2n$ times check. Thus, the prover and the verifier need 3 \heartsuit cards and $2M + 1$ \clubsuit cards in Step 2. Therefore, our protocol needs $2n^2 + 2M + 4$ cards (specifically, $n^2 + 3$ \heartsuit cards and $n^2 + 2M + 1$ \clubsuit cards) in total.

Next, we consider the number of shuffle in our protocol. Since pile-scramble shuffle is executed in Step 2(c)ii and Step 2(c)iv, $2n$ shuffles are executed in Step 2(c). In addition, Step 2(c) is executed $2n$ times in Step 2. Thus, our protocol needs $2n \times 2n = 4n^2$ shuffles.

5 Proof of Security

Here, we show our protocol satisfies the three conditions for ZKP.

5.1 Completeness

Lemma 1. If the prover knows the solution, the verifier always accepts.

Proof. If the prover knows the solution, the sum of the integers in cells, where two cards $\boxed{\heartsuit} \boxed{\clubsuit}$ are placed in this order, equals the target value for all rows and columns. Thus, the verifier does not reject for all rows and columns. Therefore, the verifier always accepts. \square

5.2 Soundness

Lemma 2. The soundness error is 0, that is, if the prover does not know the solution, the verifier always rejects.

Proof. We prove a contraposition of this lemma: if the verifier accepts, the prover knows the solution of the given instance of Sumplete. When the verifier accepts, the sum of cells where $\boxed{\heartsuit} \boxed{\clubsuit}$ are placed equals the target value for every row and column. Thus, we can see that the positions of the integers in the cells where $\boxed{\clubsuit} \boxed{\heartsuit}$ are placed are the solution. This fact implies that the prover knows the solution. Since the above argument holds with probability 1, the contraposition of Lemma 2 holds. \square

5.3 Zero Knowledge

Lemma 3. During the protocol, the verifier learns nothing about the solution of the given instance.

Proof. To prove zero-knowledge, it is sufficient that all distributions of opened cards can be simulated without the solution. In order to prove the zero-knowledge property, it is sufficient to show that all distributions of cards opened during the protocol execution can be simulated without the solution.

- In Step 2(c)iii, we open the leftmost column in the card matrix made in Step 2(c)i. Before the opening operation, we apply the pile-scramble shuffle to the two piles; one consists of the upper row, and the other consists of the lower row of the card matrix. Thus, $\boxed{\heartsuit}$ appears at each row with the same probability, i.e., probability $1/2$. Therefore, the distribution of cards opened in Step 2(c)iii can be simulated without the solution.

- In Step 2(c)v, we open the second leftmost column in the card matrix made in Step 2(c)i. Before the opening operation, we apply the pile-scramble shuffle to the two piles of the upper row and the lower row of the card matrix. Thus, $\boxed{\heartsuit}$ appears at each low with the probability $1/2$. Therefore, the distribution of cards opened in Step 2(c)v can be simulated without the solution.
- In Step 2(d), we open the true counter after adding integers in the row or column for every row and column. If the prover knows the solution, the value represented by the true counter equals the target value of the row (or column). Therefore, the distribution of the true counter's cards opened in Step 2(d) can be simulated without the solution. Specifically, the true counter represents the target value with the probability of 1.

Therefore, the verifier learns nothing about the solution. \square

6 Conclusion

In this paper, we proved that Sumplete, a puzzle generated by ChatGPT, is NP-complete and proposed a physical zero-knowledge proof protocol. In our zero-knowledge proof protocol, we realized the addition of not only positive integers but negative integers by expansion the usual technique. Moreover, we use the decoy technique to conceal the solution from the verifier.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers JP22KJ1362 and JP23KJ0968.

References

1. Boer, B.: More efficient match-making and satisfiability *the five card trick*. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 208–217. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_23
2. Bultel, X., Dreier, J., Dumas, J., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: FUN, vol. 49, pp. 8:1–8:20 (2016)
3. Bultel, X., et al.: Physical zero-knowledge proof for Makaro. In: SSS, pp. 111–125 (2018)
4. Chien, Y., Hon, W.: Cryptographic and physical zero-knowledge proof: from Sudoku to Nonogram. In: FUN, pp. 102–112 (2010)
5. Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 319–330. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_27
6. Dumas, J., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: COCOON, pp. 166–177 (2019)
7. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **18**(1), 186–208 (1989)
8. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. *Theory Comput. Syst.* **44**(2), 245–268 (2009)

9. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: UCNC, vol. 9252, pp. 215–226 (2015)
10. Komano, Y., Mizuki, T.: Card-based zero-knowledge proof protocol for pancake sorting. In: SecITC, pp. 222–239 (2022)
11. Lafourcade, P., et al.: How to construct physical zero-knowledge proofs for puzzles with a “single loop condition.” *Theor. Comput. Sci.* **888**, 41–55 (2021)
12. Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: A physical ZKP for slitherlink: how to perform physical topology-preserving computation. In: ISPEC, pp. 135–151 (2019)
13. Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: Practical card-based implementations of yao’s millionaire protocol. *Theor. Comput. Sci.* **803**, 207–221 (2020)
14. Miyahara, D., et al.: Card-based ZKP protocols for Takuzu and Juosan. In: FUN, pp. 20:1–20:21 (2021)
15. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for Kakuro. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **102-A(9)**, 1072–1078 (2019)
16. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: FAW, pp. 358–369 (2009)
17. Nakai, T., Tokushige, Y., Misawa, Y., Iwamoto, M., Ohta, K.: Efficient card-based cryptographic protocols for millionaires’ problem utilizing private permutations. In: CANS, pp. 500–517 (2016)
18. OpenAI: GPT-4 technical report (2023)
19. Penguin, P.: ChatGPT invented its own puzzle game (2023). <https://puzzledpenguin.substack.com/p/chatgpt-invented-its-own-puzzle-game>
20. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for sudoku. *New Gener. Comput.* **40(1)**, 49–65 (2022)
21. Ruangwises, S.: An improved physical ZKP for nonogram and nonogram color. *J. Comb. Optim.* **45(5)**, 122 (2023)
22. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. In: TAMC, pp. 25–36 (2020)
23. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Numberlink. In: FUN, vol. 157, pp. 22:1–22:11 (2021)
24. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for numberlink puzzle and k vertex-disjoint paths problem. *New Gener. Comput.* **39(1)**, 3–17 (2021)
25. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Ripple Effect. In: WALCOM, vol. 12635, pp. 296–307 (2021)
26. Ruangwises, S., Itoh, T.: How to physically verify a rectangle in a grid: a physical ZKP for shikaku. In: FUN, pp. 24:1–24:12 (2022)
27. Ruangwises, S., Itoh, T.: Physical ZKP for makaro using a standard deck of cards. In: TAMC, vol. 13571, pp. 43–54 (2022)
28. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for sudoku. *Theor. Comput. Sci.* **839**, 135–142 (2020)
29. Sasaki, T., Mizuki, T., Sone, H.: Card-based zero-knowledge proof for sudoku. In: FUN, vol. 100, pp. 29:1–29:10 (2018)
30. Shinagawa, K., et al.: Multi-party computation with small shuffle complexity using regular polygon cards. In: ProvSec, pp. 127–146 (2015)
31. Shinagawa, K., et al.: Card-based protocols using regular polygon cards. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **100-A(9)**, 1900–1909 (2017)
32. Takashima, K., et al.: Card-based protocols for secure ranking computations. *Theor. Comput. Sci.* **845**, 122–135 (2020)