Card-based zero-knowledge proof for the nearest neighbor property: Zero-knowledge proof of ABC end view

Takuro Fukasawa[1] and Yoshifumi Manabe[1 [0000−0002−6312−257X]]

Kogakuin University, Shinjuku,Tokyo 163–8677 Japan.
manabe@cc.kogakuin.ac.jp

Abstract. This paper shows a zero-knowledge proof protocol of a solution to ABC end view puzzle using physical cards. Card-based cryptographic protocols are proposed to execute a secure multi-party calculation using physical cards instead of computers. This paper shows a card-based zero-knowledge proof of the ABC end view puzzle. The puzzle needs a new technique to prove the nearest neighbor from an end. We show a new zero-knowledge proof protocol to securely calculate the nearest neighbor using physical cards.

Keywords: Card based cryptographic protocols · Zero-knowledge proof · ABC end view · nearest neighbor.

1 Introduction

This paper shows a zero-knowledge proof protocol of a solution of ABC end view puzzle[6] using physical cards. Card-based cryptographic protocols[2, 18] are proposed to execute a secure multi-party calculation using physical cards instead of computers. These protocols can be used when the users cannot trust the software on the computer. Many protocols were shown to calculate any boolean functions[10, 13, 32] and specific problems such as voting[1, 17] and millionaires' problem[14, 20, 21] and so on.

As another usage of card-based cryptographic protocols, zero-knowledge proof of puzzle solutions was proposed. The protocol proves that a user has a solution to the puzzle without leaking any information about the solution.

A zero-knowledge proof of Sudoku [8] was first considered. The proof has a soundness error, thus improved zero-knowledge proofs were shown[26, 31]. Zero-knowledge proofs of the other puzzles are shown, for example, Akari[3], Flow Free[9], Heyawake[23], Hitori[23], Juosan[15], Kakuro[3, 16], KenKen[3], Makaro[4, 29], Masyu[12], Nonogram[5, 25], Norinori[7], Numberlink[27], Nurikabe[23], Nurimisaki[24], Ripple Effect[28], Shikaku[30], Slitherlink[12], Suguru[22], Takuzu[3, 15], Topswops[11], and so on.

This paper shows a zero-knowledge proof of the ABC end view puzzle. The proof needs a new technique to prove the nearest neighbor from an end. We show a new protocol to securely calculate the nearest neighbor using physical cards.

Section 2 shows the problem definition. Section 3 shows the protocol. Section 4 concludes the paper.

2 Definition of problem

A zero-knowledge proof for a language L is a protocol executed by two players, called prover P and verifier V. The prover has an element x.

- (Completeness) If $x \in L$, an honest verifier V is convinced that $x \in L$ by an honest prover P.
- (Soundness) If $x \notin L$, no cheating prover P can convince an honest verifier V that $x \in L$.
- (Zero-knowledge) If $x \in L$, no verifier V learns anything other than the fact that $x \in L$.

For the problem of a solution to a puzzle, if the prover has a solution, an honest verifier is convinced that the prover has a solution. If the prover does not have a solution, the prover cannot convince an honest verifier that the prover has a solution. By the execution of the protocol, the verifier has no information about the solution.

ABC end view (aka "Easy as ABC" or "Last man standing") [6] is a pencil puzzle. The problem is given as a grid and a range of letters, for example, A-E. Each different letter must occur exactly once in each row and column. The letters outside the grid show which letter comes across first from that direction. An example of the problem of a 5*5 grid and range A-C is shown in the left of Fig. 1. The solution to the problem is shown on the right of Fig. 1, where "×" means no letter is written in the space.

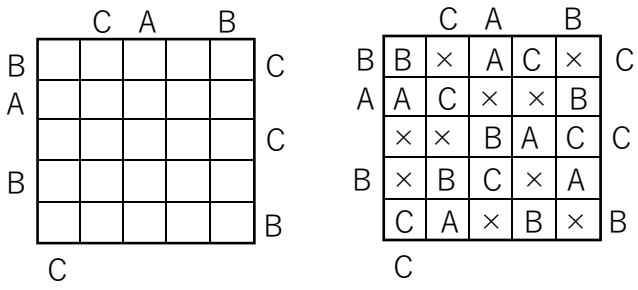

Fig. 1. An example of ABC end view problem and its solution

Card-based cryptographic protocols use physical cards to securely calculate values. For calculations of boolean functions, two kinds of cards, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ are used. Cards of the same marks cannot be distinguished. In addition, the back of

both types of cards is $\boxed{?}$. It is impossible to determine the mark on the back of a given card of $\boxed{?}$. Some additional cards are used for the zero-knowledge proof protocols. The first type of card is the number card, whose marks are $\boxed{1}$, $\boxed{2}$, ..., $\boxed{n}$, where n is the size of the grid. The second type of card is the letter card, whose marks are $\boxed{A}$, $\boxed{B}$, $\boxed{C}$, and so on. The last type of card is the empty card, whose mark is $\boxed{\times}$, which means "no letter". Cards of the same marks cannot be distinguished. In addition, the back of all types of cards is $\boxed{?}$. Note that any kind of card can be represented by an appropriate encoding using the two kinds of cards $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. For example, a number card can be represented by $\lceil \log n \rceil$ $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ cards. Each bit of the number is represented by the encoding rule $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. For the letter cards and the empty card, a similar encoding rule can be introduced and one card is represented by several numbers of $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ cards. For the simplicity of the discussion, this paper uses additional cards.

3 Protocol for ABC end view

Zero-knowledge proof of a solution to an ABC end view puzzle is executed as follows. First, the prover P sets the solution of the given puzzle in a committed manner, that is, V cannot see the values of the solution. Then P and V execute the verification protocol to prove the solution is correct without knowing the values. They need to prove the following two properties

- nearest neighbor property: The letter in the grid that is nearest to the letter written outside of the grid must be correct.
- uniqueness property: Every row and column has just one letter for the given range of letters.

For an example of the uniqueness property, if the range is A-C, A, B, and C appear once in the squares of each row and column. All the other squares have $\times$ as the example in Fig. 1.

Initially, we show card-based cryptographic protocols used in this paper. One-bit data is represented by two cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$.

One pair of cards that represents one bit $x \in \{0, 1\}$, whose face is down, is called a commitment of x, and denoted as $commit(x)$. It is written as $\underbrace{\boxed{?}\boxed{?}}_{x}$. Note that when these two cards are swapped, $commit(\bar{x})$ can be obtained. Thus, logical negation can be easily calculated.

A set of cards placed in a row is called a sequence of cards. A sequence of cards S whose length is n is denoted as $S = s_1, s_2, \ldots, s_n$, where s_i is i-th card of the sequence. $S = \underbrace{\boxed{?}}_{s_1}\underbrace{\boxed{?}}_{s_2}\underbrace{\boxed{?}}_{s_3}\ldots\underbrace{\boxed{?}}_{s_n}$.

A shuffle is executed on a sequence of cards S. Its parameter is $(\Pi, \mathcal{F})$, where Π is a set of permutations on S and $\mathcal{F}$ is a probability distribution on Π. For a

given sequence S, each permutation $\pi \in \Pi$ is selected by the probability distribution $\mathcal{F}$ and π is applied to S. If π is applied on $S = s_1, s_2, \ldots, s_n$, the result is $s_{\pi^{-1}(1)}, s_{\pi^{-1}(2)}, \ldots, s_{\pi^{-1}(n)}$. Since π is selected from Π, the result is not deterministic. Non-deterministic shuffles are necessary for card-based cryptographic protocols to make the protocols secure at opening cards. As shown in the below protocol, cards on each row are randomly shuffled and then opened to show that A, B, and C appear once. If the shuffle is deterministic, the players know the initial position where the A card was set as the answer. Therefore, a random shuffle whose result is unknown to the players is necessary.

We show examples of shuffles used in the protocols shown below. A random shuffle is randomly changing the positions of the cards for the given sequence of cards. When $S = s_1, s_2, s_3$, the result of a random shuffle is $S_1 = s_1, s_2, s_3$, $S_2 = s_1, s_3, s_2$, $S_3 = s_2, s_1, s_3$, $S_4 = s_2, s_3, s_1$, $S_5 = s_3, s_1, s_2$, or $S_6 = s_3, s_2, s_1$. The probability of obtaining each result is $1/|S|!$.

A random bisection cut is swapping the left half and the right half of a given even-length sequence. When $S = s_1, s_2, s_3, s_4, s_5, s_6$, the result of a random bisection cut is $S_0 = s_1, s_2, s_3, s_4, s_5, s_6$ or $S_1 = s_4, s_5, s_6, s_1, s_2, s_3$. The probability of obtaining each result is $1/2$. The random bisection cut is considered as selecting a random bit $b \in \{0, 1\}$ and obtaining S_b.

Next, we introduce piles of cards. A pile of cards is a sequence of cards whose order cannot be changed using some additional tools such as clips or envelopes. For example, consider a case when cards $x_{i,j} (i = 1, 2, \ldots, n, j = 1, 2, \ldots m)$ are given. The players make piles of cards such that $plie_i = x_{i,1}, \ldots, x_{i,m} (i = 1, 2, \ldots, n)$ using clips or envelopes. The players treat each pile $pile_i$ just like a single card during shuffle operations. The order of cards in a pile cannot be changed because of the clip or envelope. For a pile y, let $y(i)$ be i-th card in y. Players can rearrange piles by removing clips, setting new sequences of cards, and making new piles. Let $y[2-]$ be the new pile that $y(1)$ is removed from y.

The shuffles can be executed for piles of cards. Consider the case shuffle π is executed on the above piles $pile_i (i = 1, 2, \ldots, n)$. The result is $pile_{\pi^{-1}(1)}, pile_{\pi^{-1}(2)}$, $\ldots, pile_{\pi^{-1}(n)}$, where $pile_{\pi^{-1}(i)} = x_{\pi^{-1}(i),1}, x_{\pi^{-1}(i),2}, \ldots, x_{\pi^{-1}(i),m}$. Random shuffles on piles are called pile-scramble shuffles.

Next, we show logical AND and copy protocols used in this paper.

Protocol 1 *(AND protocol)[19]*
Input: commit(x) and commit(y).
Output: commit$(x \wedge y)$.

1. *Input commit(x) and commit(y) are set as Fig. 2 (a).*
2. *The positions of the cards are changed as Fig. 2 (b).*
3. *Execute a random bisection cut on the sequence of the cards. The result can be written as follows: select a random bit $b \in \{0, 1\}$, that is unknown to the players. If $b = 0$, there is no change in the order of the cards. If $b = 1$, the left half and the right half are swapped as Fig. 2 (c).*
4. *Change the sequence of the cards as Fig. 2 (d).*

5. *Open the left two cards. If the sequence is* ♣♡*, the center two cards are* $commit(x \wedge y)$*. Otherwise, the right two cards are* $commit(x \wedge y)$*, as Fig. 2 (e).*

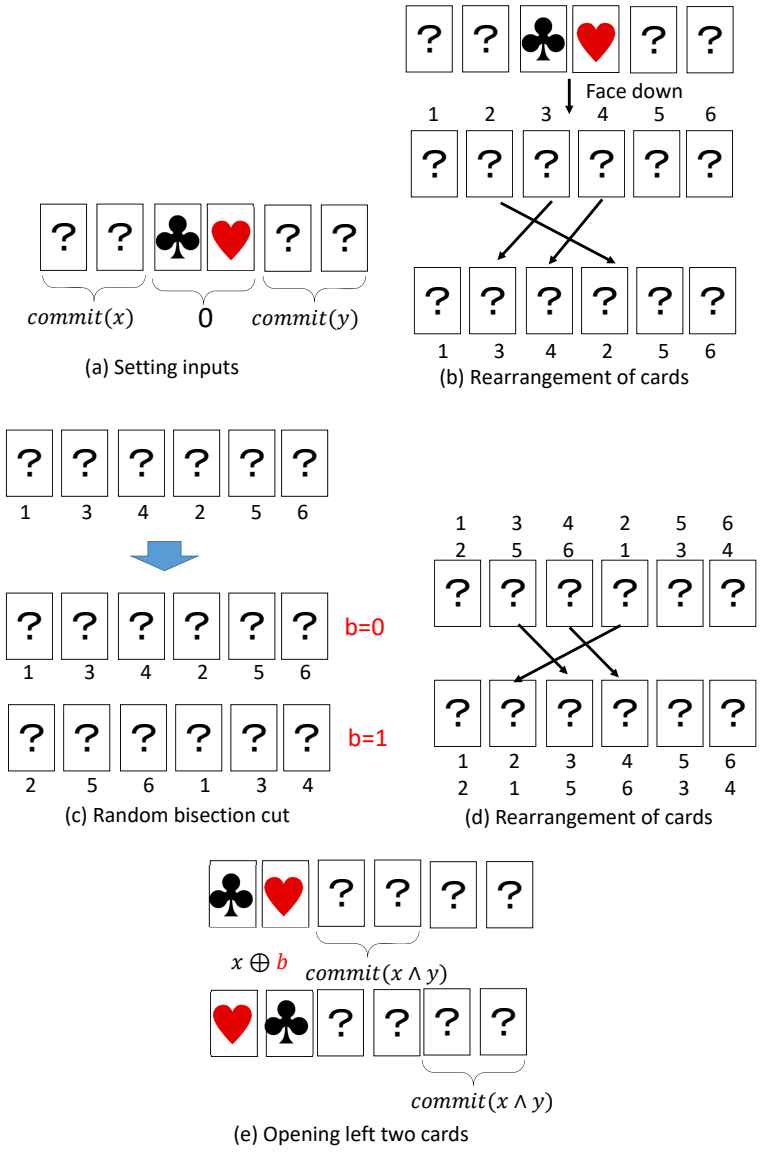

Fig. 2. AND protocol in [19] .

The protocol outputs

$$x \wedge y = \begin{cases} y & \text{if } x = 1 \quad (\boxed{\heartsuit}, \boxed{\clubsuit}) \\ 0 & \text{if } x = 0 \quad (\boxed{\clubsuit}, \boxed{\heartsuit}) \end{cases} \tag{1}$$

That is, the output is the left two cards if $x = 1$. The output is the center two cards if $x = 0$. The protocol opens two cards $x \oplus b$. Since b is a random number unknown to the players, the security of the private input data is achieved. The detailed proof is shown in [19].

Next, we show copy protocol, which gives multiple copies of a given input commitment.

Protocol 2 *(copy protocol using random bisection cuts)[19]*

Input: commit(x)*.*

Output: two copies of commit(x)*.*

1. *Input commit*(x) *and two copies of commit*(0) *are set as Fig. 3 (a).*
2. *The positions of the cards are changed as Fig. 3 (b).*
3. *Execute a random bisection cut on the sequence of the cards. The result can be written as follows: select a random bit* $b \in \{0, 1\}$*, that is unknown to the players. If* $b = 0$*, there is no change in the order of the cards. If* $b = 1$*, the left half and the right half are swapped as Fig. 3 (c).*
4. *Change the sequence of the cards as Fig. 3 (d).*
5. *Open the left two cards. If the sequence is* $\boxed{\clubsuit}\boxed{\heartsuit}$*, the remaining pairs are commit*(x)*. Otherwise, the remaining pairs are commit*$(\bar{x})$*, as Fig. 3 (e). In this case, commit*(x) *can be obtained by swapping the two cards of commit*$(\bar{x})$*.*

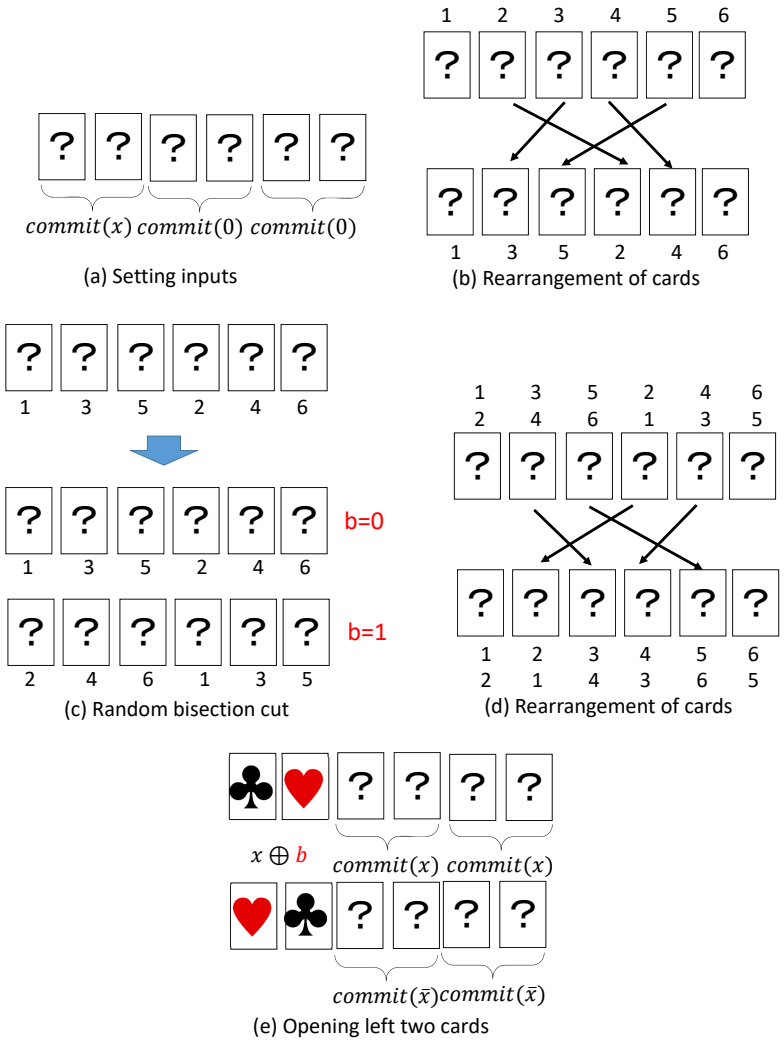

Fig. 3. Copy protocol in [19].

We show the zero-knowledge proof protocol for ABC end view in Algorithms 1-3. Algorithm 1 is the main routine and Algorithm 2 is the subroutine to verify the nearest neighbor property. Algorithm 3 is the subroutine to verify the uniqueness property. In the following protocol description, the corresponding code at Line j of Algorithm i is written as "(L. $j(i)$)". The outline of the protocol is as follows. Suppose that the grid is $n * n$ and the number of letters is c. In the example in Fig. 1, $n = 5$ and $c = 3$. First, the prover P sets the solution

in a committed manner as follows. For the square at i-th row and j-th column (denoted as square (i, j)), P puts face-down $\boxed{\heartsuit}$, $\boxed{\clubsuit}$, and $\boxed{\mathbf{L}}$ in this order if the solution is letter L. P puts face-down $\boxed{\clubsuit}$, $\boxed{\heartsuit}$, and $\boxed{\times}$ in this order if the solution is "no letter" (L. 3-5(1)). Thus the sequence $\boxed{\heartsuit}$, $\boxed{\clubsuit}$ means the solution is a letter. These cards are denoted as $x_{i,j(1)}$, $x_{i,j(2)}$, and $x_{i,j(3)}$ (L. 6(1)).

Algorithm 1 Zero knowledge proof protocol of ABC end view

```
 1: procedure MAIN
 2:     Let k = n − c + 1
 3:     At square (i, j), P sets
 4:     face-down ♡, ♣, and L in this order if the solution is letter L,
 5:     face-down ♣, ♡, and × in this order if the solution is 'no letter'.
 6:     These cards are denoted as x_{i,j}(1), x_{i,j}(2), and x_{i,j}(3).
 7:     for i=1 to n do
 8:         for j=1 to n do
 9:             P and V put face-up card j to the right of three cards on square (i, j).
10:             They face down the card. The card is denoted as x_{i,j}(4).
11:             The pile of cards at (i, j) is denoted as x_{i,j}.
12:         end for
13:         if There is a letter at the left end of i-th row then
14:             Execute nearestneighbor.
15:         end if
16:         if There is a letter at the right end of i-th row then
17:             Execute nearestneighbor.
18:         end if
19:         Execute uniqueness at i-th row.
20:         Face-down x'_j(1), x'_j(2), and x'_j(3) for each pile x'_j.
21:         Execute pile scramble shuffle on x'_1, ..., x'_n.
22:         Let the results be x''_1, ..., x''_n.
23:         Open x''_j(4) of each pile.
24:         If x''_j(4) = l, put x''_j(1), x''_j(2), and x''_j(3) to square (i, l).
25:     end for/* Each row check is finished. */
26:     for i=1 to n do
27:         if There is a letter at the top end of i-th column then
28:             Execute nearestneighbor.
29:         end if
30:         if There is a letter at the bottom of i-th column then
31:             Execute nearestneighbor.
32:         end if
33:         Execute uniqueness at i-th column.
34:     end for/* Each column check is finished. */
35: end procedure
```

P and V set number cards to remember the positions of the cards, since the positions are changed by the verification. P and V publically put face-up card $\boxed{\mathbf{j}}$ to the right of three cards on square (i, j) and then face down the card (L.

Algorithm 2 Subroutine: nearest neighbor verification

procedure NEARESTNEIGHBOR
  Select k plies on the squares in the current row or column.
  Let $y_i (1 \leq i \leq k)$ be the piles, where y_1 is the closest to the end.
  Let z be y_k.
  for $j = k - 1$ downto 1 **do**
    Execute copy protocol on $(y_j(1), y_j(2))$.
    Let the obtained pair be $(y'_j(1), y'_j(2))$.
    Execute AND protocol using $(y'_j(1), y'_j(2), y_j(1), y_j[2-], z(1), z[2-])$.
    if The left two opened cards are ($\boxed{\clubsuit}$,$\boxed{\heartsuit}$) **then**
      Set the right card-pile pair as new z.
      Set the center card-pile pair as new y_{j+1}.
    else
      Set the center card-pile pair as new z.
      Set the right card-pile pair as new y_{j+1}.
    end if
  end for
  Open $z(3)$ and verify that the letter is the same as the one written outside.
  Face-down $z(3)$ and set pile z to new y_1.
  Put $y_1, y_2, \ldots, y_k$ to the original squares.
end procedure

Algorithm 3 Subroutine: uniqueness verification

procedure UNIQUENESS
  Let $x_1, \ldots, x_n$ be the piles in the current row or column.
  Execute pile scramble shuffle on $x_1, \ldots, x_n$.
  Let the results be $x'_1, \ldots, x'_n$.
  Open $x'_j(1)$, $x'_j(2)$, and $x'_j(3)$ of each pile x'_j.
  Verify that (1) if $x'_j(3)$ is a letter, $(x'_j(1), x'_j(2))$ is ($\boxed{\heartsuit}$,$\boxed{\clubsuit}$), otherwise $(x'_j(1), x'_j(2))$ is ($\boxed{\clubsuit}$,$\boxed{\heartsuit}$) and (2) all letters on the letter cards differ from each other and the number of $\boxed{\times}$ cards is $n - c$.
end procedure

9(1)). The card is denoted as $x_{i,j(4)}$(L.10(1)). The four cards form a pile $x_{i,j}$ (L. 11(1)).

First, we show the procedure to verify the neighborhood property(called at L. 14(1), 17(1), 28(1), and 31(1)). Let i be the current row to verify and consider the case when there is a letter L at the left end. Let $k = n - c + 1$. The candidate of the nearest square that has a letter is $(i, 1), (i, 2) \dots (i, k)$, since the number of "no letter" squares is $n - c$ (L. 2(2)).

The procedure to obtain the nearest neighbor is as follows:

1. Let $y_j = x_{i,j} (1 \leq j \leq k-1)$ and $z = y_k$.
2. For $j = k - 1$ down to 1 Do
3. If $(y_j(1), y_j(2)) = (\boxed{\heartsuit}, \boxed{\clubsuit})$ then $z = y_j$
4. EndFor
5. Return z

Initially, the candidate z is y_k, the farthest from the border (L. 4(2)). If the nearer square has a letter (that is, $(y_j(1), y_j(2)) = (\boxed{\heartsuit}, \boxed{\clubsuit})$), replace the candidate. After the test at $j = 1$ is finished, z has the nearest letter. For example, consider the case when the players verify the left end "B" in the first row in Fig. 1. Since $k = 3$, the players set $y_i = x_{1,i} (1 \leq i \leq 2)$, and $z = x_{1,3}$. Execute the for loop and when $j = 2$, $(y_j(1), y_j(2)) = (\boxed{\clubsuit}, \boxed{\heartsuit})$ and z is unchanged. When $j = 1$, $(y_j(1), y_j(2)) = (\boxed{\heartsuit}, \boxed{\clubsuit})$ and $z = y_1$ is executed. Thus the final $z = y_1$ and the letter is "B".

We need to execute the above procedure without knowing the value $(y_j(1), y_j(2))$. We use AND protocol to solve the problem. The if statement in step 3 can be written as follows:

$$new\ z = \begin{cases} y_j & \text{if } y_j = 1 \quad (\boxed{\heartsuit}, \boxed{\clubsuit}) \\ z & \text{if } y_j = 0 \quad (\boxed{\clubsuit}, \boxed{\heartsuit}) \end{cases} \tag{2}$$

Comparing this equation and Equation (1), we can obtain the result using AND protocol if we set $y_j(1), y_j(2), z(1), z[2-], y_j(1), y_j[2-]$ in this order[1] at the first step in Fig. 2(a) (L. 8(2)). We need a copy of $(y_j(1), y_j(2))$, thus the copy protocol is executed to $(y_j(1), y_j(2))$ in advance (L. 6(2)). Another difference between the AND protocol is that $z[2-]$ and $y_j[2-]$ are a pile of cards. This change does not reveal the secret random value b of the AND protocol, because the positions of piles are the fourth and sixth positions in Fig. 2(b) and (c), The piles come to the same positions when $b = 0$ and $b = 1$. Thus, the difference between a card and a pile does not reveal the secret random value b.

When a new z is selected, the unused pile (old z or y_j) is put to the square $(i, j + 1)$ for further verification from the other side(L. 11(2), L. 14(2)).

[1] It is unnecessary to divide the piles as $z(1)$ and $z[2-]$. We set $y_j(1), y_j(2), z, y_j$ in this order, swap the second and the third, execute a random bisection cut, swap the second and the third, open the left two cards, and obtain the result as the AND protocol. The result is the same as the protocol shown in this paper.

When the procedure is finished, the players open $z(3)$ to see the final result (L. 17(2)). If the letter on the card is the same as the letter written outside of the grids, the verification succeeds.

The final z is put to the square $(i, 1)$ (L.18(2)). Though the positions of the piles differ from the initial position set by P, the change does not affect the next verification from the other side. The reason is as follows. Pile z with some letter might go left (to the position of the smaller index), but it will not go left further to the position where another letter exists. If y_j has a letter, new z becomes y_j, and old z is put to the position of $(i, j + 1)$. Thus, the relative order of the letters in the i-th row does not change. For example, consider the case when the players verify the left end "B" in the first row in Fig. 1. After the verification, the letters in the first row become "B A × C ×", but the relative order of "B" and "A" is not changed. Thus, the verification for the right end works with this modified sequence.

After the right and left nearest neighbor verifications of i-th row, uniqueness verification of i-th row is executed(L. 19(1)). Note that the position of each pile differs from the initial positions P set, but the change does not affect the verification since the set of letters in i-th row does not change during the nearest neighbor verifications.

The verification technique is just the same as the one used for Sudoku [31]. Execute a pile-scramble shuffle to the piles $x_{i,1}, \ldots, x_{i,n}$ in i-th row(L. 3(3)). Let the results be $x'_1, x'_2, \ldots, x'_n$ (L. 4(3)). Then open $x'_j(l)(1 \leq j \leq n, 1 \leq l \leq 3)$ (L. 5(3)). V verifies that all the letters in $x'_j(3)$ differ and the number of $\boxed{\times}$ in $x'_j(3)$ is $n - c$. In addition, V checks the consistency of cards, that is, $(x'_j(1), x'_j(2)) =$ ($\boxed{\heartsuit}$,$\boxed{\clubsuit}$) or ($\boxed{\clubsuit}$,$\boxed{\heartsuit}$) must be satisfied. If $(x'_j(1), x'_j(2)) =$ ($\boxed{\heartsuit}$,$\boxed{\clubsuit}$), $x'_j(3)$ must be a letter card, otherwise $x'_j(3)$ must be $\boxed{\times}$ (L. 6(3)).

After the verification is finished, P and V face-down $x'_j(l)(1 \leq j \leq n, 1 \leq l \leq 3)$ (L. 20(1)). Then the players execute a pile-scramble shuffle on $x'_1, x'_2, \ldots, x'_n$ (L. 21(1)). Let the results be $x''_1, x''_2, \ldots, x''_n$ (L.22(1)). Then open $x''_j(4)(1 \leq j \leq n)$ (L.23(1)). If $x''_j(4) = l$, put x''_j to square (i, l) (L.24(1)). Each pile is moved to the original square P set. Note that the number cards are no more necessary for the verifications of each column because it is unnecessary to move the plies to the original squares again.

For each column, execute the above nearest neighbor check and uniqueness check (L.26-34(1)). The only difference is each pile consist of three cards. Since the number cards are used only for resetting the positions of piles and are not used in the verification itself, there is no change in the procedure.

(Example) We show the steps for the first row of the problem in Fig. 1. P sets the solution by face-down cards as in Fig. 4 (a). Since the solution is "B × A C ×", P sets $(x_{1,1}(1), x_{1,1}(2), x_{1,1}(3)) =$ ($\boxed{\heartsuit}$,$\boxed{\clubsuit}$,$\boxed{\mathbf{B}}$), $(x_{1,2}(1), x_{1,2}(2), x_{1,2}(3)) =$ ($\boxed{\clubsuit}$,$\boxed{\heartsuit}$,$\boxed{\times}$), $(x_{1,3}(1), x_{1,3}(2), x_{1,3}(3)) =$ ($\boxed{\heartsuit}$,$\boxed{\clubsuit}$,$\boxed{\mathbf{A}}$), $(x_{1,4}(1), x_{1,4}(2), x_{1,4}(3)) =$ ($\boxed{\heartsuit}$,$\boxed{\clubsuit}$,$\boxed{\mathbf{C}}$), and $(x_{1,5}(1), x_{1,5}(2), x_{1,5}(3)) =$ ($\boxed{\clubsuit}$,$\boxed{\heartsuit}$,$\boxed{\times}$) in the face-down manner (Note that in Fig. 4, the sequence of cards at each square are written from the top to bottom).

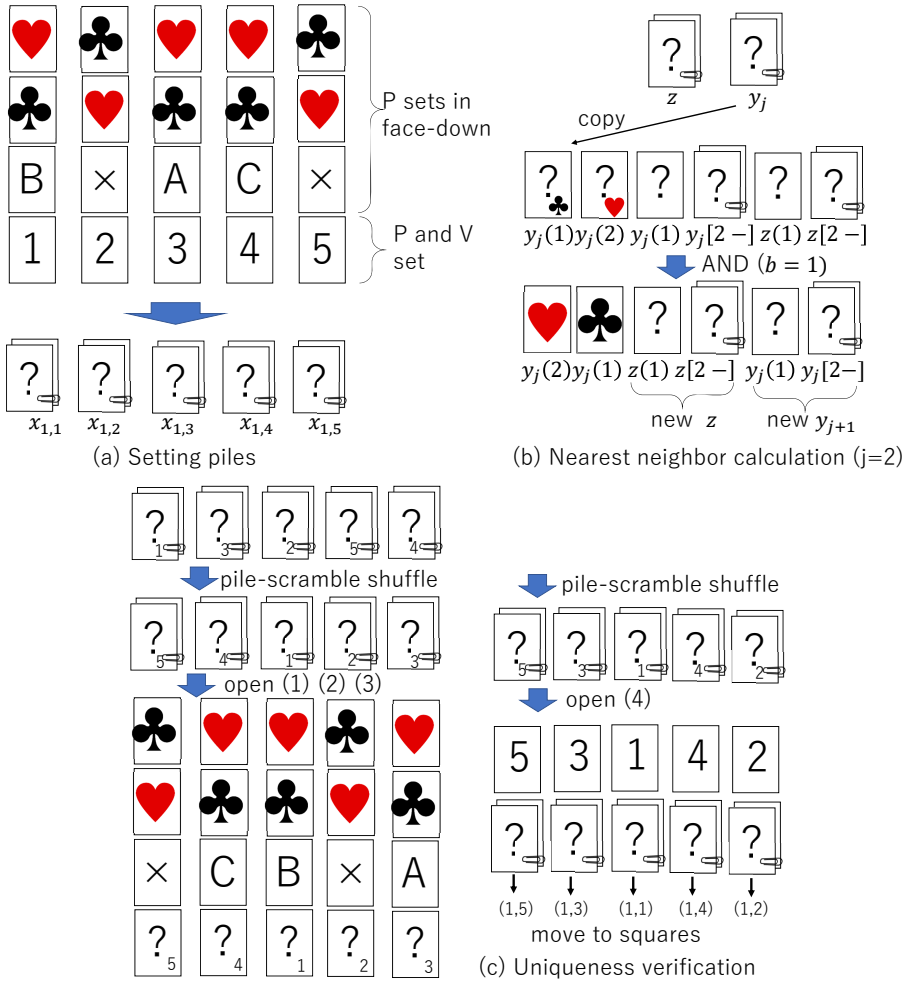

Fig. 4. Example of execution in the first row.

Then P and V sets $x_{1,1}(4) = \boxed{\mathbf{1}}$, $x_{1,2}(4) = \boxed{\mathbf{2}}$, $x_{1,3}(4) = \boxed{\mathbf{3}}$, $x_{1,4}(4) = \boxed{\mathbf{4}}$, and $x_{1,5}(4) = \boxed{\mathbf{5}}$. These cards are turned face-down. They make piles $x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4}$ and $x_{1,5}$ as Fig. 4 (a). Then they execute the nearest neighbor verification algorithm. Since $n = 5$ and $c = 3$, $k = n - c + 1 = 3$. Thus, the players select three piles from the left end. $y_1 = x_{1,1}$, $y_2 = x_{1,2}$, and $y_3 = x_{1,3}$. Then set $z = y_3$. First, the modified AND protocol is executed between y_2 and z, as shown in Fig. 4(b). Execute the copy protocol on $(y_2(1), y_2(2))$. Note that since the cards are face-down, the marks of the cards are unknown, but they are written in Fig. 4(b) by small marks for the explanation. In this example,

$(y_2(1), y_2(2)) = (\boxed{\clubsuit}, \boxed{\heartsuit})$. Set sequence $y_2(1), y_2(2), y_2(1), y_2[2-], z(1), z[2-]$ and execute AND protocol. Suppose that $b = 1$ by the random bisection cut. In the case, the final sequence becomes $y_2(2), y_2(1), z(1), z[2-], y_2(1), y_2[2-]$. The left two cards are opened. Since they are $(\boxed{\heartsuit}, \boxed{\clubsuit})$, the center two elements are selected as the new z. The left two elements are set as y_3. In this case, z remains at the position of z and y_2 is moved to y_3 as in Fig. 4(b).

Next, the players execute the modified AND protocol between z and y_1. Since $(y_1(1), y_1(2)) = (\boxed{\heartsuit}, \boxed{\clubsuit})$, y_1 becomes the new z. Old z is moved to y_2. The final value of z is obtained. Since it is y_1, $\boxed{\mathbf{B}}$ appears when $z(3)$ is opened. Thus, the verification succeeds. z is then set at the position of y_1. The piles are then moved to the squares. Thus, the letter cards of $(x_{1,1}, x_{1,2}, x_{1,3})$ are changed as $(\boxed{\mathbf{B}}, \boxed{\mathbf{A}}, \boxed{\times})$.

Next, the players verify the right end card. $k = 3$ and $y_1 = x_{1,5}$, $y_2 = x_{1,4}$ and $y_3 = x_{1,3}$. Note that the letter card of $x_{1,3}$ is currently $\boxed{\times}$ by the above procedure. Initially, $z = y_3$ and execute modified AND protocol between z and y_2. Since $(y_2(1), y_2(2)) = (\boxed{\heartsuit}, \boxed{\clubsuit})$, y_2 becomes the new z. Old z is set as y_3. Next, the modified AND protocol is executed between z and y_1. Since $(y_1(1), y_1(2)) = (\boxed{\clubsuit}, \boxed{\heartsuit})$, z remains unchanged. y_1 is set as new y_2. Thus, when $z(3)$ is opened, the card is $\boxed{\mathbf{C}}$ and the letter is correct.

Then the uniqueness verification is executed. After the nearest neighbor verifications, the letter cards of $(x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4}, x_{1,5})$ are changed as $(\boxed{\mathbf{B}}, \boxed{\mathbf{A}}, \boxed{\times}, \boxed{\times}, \boxed{\mathbf{C}})$, that is, the number cards are $(\boxed{\mathbf{1}}, \boxed{\mathbf{3}}, \boxed{\mathbf{2}}, \boxed{\mathbf{5}}, \boxed{\mathbf{4}})$. Note that in Fig 4(c), the numbers are written in a small font for the explanation, but the players cannot see the cards. Then the players execute a pile-scramble shuffle. The order of the piles is randomly changed. Suppose that the order is changed as $(5, 4, 1, 2, 3)$ as Fig. 4(c). Let the result as $x'_1, x'_2, x'_3, x'_4, x'_5$. The players open $x_j(1), x_j(2)$, and $x_j(3)$ for every $j (j = 1, 2, \ldots, 5)$. The players verify that the numbers of each card of $\boxed{\mathbf{A}}$, $\boxed{\mathbf{B}}$, and $\boxed{\mathbf{C}}$ are one. Pile x_i with a letter card has $(x_i(1), x_i(2)) = (\boxed{\heartsuit}, \boxed{\clubsuit})$. Pile x_j with $\boxed{\times}$ card has $(x_j(1), x_j(2)) = (\boxed{\clubsuit}, \boxed{\heartsuit})$. Thus the uniqueness verification is finished. The players face down the opened cards and make piles again. The players execute a pile-scramble shuffle again. Let the result be $x''_1, x''_2, x''_3, x''_4, x''_5$. The players open $x''_j(4)$ for every $j (j = 1, 2, \ldots, 5)$. If $x''_j(4) = i$, $x''_j(1), x''_j(2), x''_j(3)$ are moved to square $(1, i)$. Each pile is moved to the original square P set as in Fig. 4(c). The piles are used for the verification of each column.

The number of cards used by the algorithm is as follows: $n^2 + 4$ cards for each of $\boxed{\heartsuit}$ and $\boxed{\clubsuit}$, n cards for each of $\boxed{\mathbf{A}}\boxed{\mathbf{B}}, \ldots, \boxed{\mathbf{L}}$ card when the range of letters is $A - L$. $n^2 - n * c$ cards of $\boxed{\times}$ and one card for each of $\boxed{\mathbf{1}}, \boxed{\mathbf{2}}, \ldots, \boxed{\mathbf{n}}$. Thus the total number of cards is $3n^2 + n + 8$.

Last, we show the correctness of the protocol.

Theorem 1. *The procedure is a zero-knowledge proof of solutions to the ABC end view problem.*

(Proof)

(Completeness) When P has a solution to the given problem and correctly sets the cards, the nearest neighbor verification and uniqueness verification succeeds in each row and column as shown above.

(Soundness) When P sets the cards that are not a solution, the fact can be detected by V. The reason is as follows. At the uniqueness verification, all cards in a row or a column are simultaneously opened. Thus, if the cards are not correct, the fact can be detected by V. In addition, the nearest neighbor verification protocol outputs the letter nearest to the end, thus if the letter is not correct, V can detect that.

(Zero-knowledge) During the uniqueness verification, V gets no information other than the fact that each row and column has one letter for each A-L. During the nearest neighbor verification, V gets no information other than the nearest letter is correct. The protocol uses AND protocol and copy protocol. They leak no information from the opened cards, as shown in [19].

During the uniqueness verification protocol, all cards in a row or a column are opened. However, they leak no information since the positions are randomized by the pile-scramble shuffles. □

4 conclusion

This paper showed a card-based zero-knowledge proof of a solution of the ABC end view puzzle. The nearest neighborhood calculation is a new technique to solve the problem. Zero-knowledge proof to the other puzzle problems is one of the further studies.

References

1. Abe, Y., Nakai, T., Kuroki, Y., Suzuki, S., Koga, Y., Watanabe, Y., Iwamoto, M., Ohta, K.: Efficient card-based majority voting protocols. New Generation Computing **40**(1), 173–198 (2022)
2. den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Proc. of EUROCRYPT '89, LNCS Vol. 434. pp. 208–217 (1990)
3. Bultel, X., Dreier, J., Dumas, J.G., Lafourcade, P.: Physical zero-knowledge proofs for akari, takuzu, kakuro and kenken. In: Proc. of 8th International Conference on Fun with Algorithms. vol. 49, pp. 8–1. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2016)
4. Bultel, X., Dreier, J., Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Nagao, A., Sasaki, T., Shinagawa, K., Sone, H.: Physical zero-knowledge proof for makaro. In: Proc. of 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2018), LNCS Vol.11201. pp. 111–125 (2018)
5. Chien, Y.F., Hon, W.K.: Cryptographic and physical zero-knowledge proof: from sudoku to nonogram. In: Proc. of International Conference on Fun with Algorithms. pp. 102–112. Springer (2010)
6. crossa@list.ru: Puzzles: Abc end view. http://www.cross-plus-a.com/puzzles.htm#EasyAsABC, (Accessed on 8/31/2022)

7. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for norinori. In: Proc. of 25th International Computing and Combinatorics Conference(COCOON 2019), LNCS Vol. 11653. pp. 166–177. Springer (2019)
8. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. Theory of Computing Systems **44**(2), 245–268 (2009)
9. Hart, E., McGinnis, J.A.: Physical zero-knowledge proofs for flow free, hamiltonian cycles, and many-to-many k-disjoint covering paths. arXiv preprint arXiv:2202.04113 (2022)
10. 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: Proc. of Asiacrypt 2017, Part III, LNCS Vol. 10626. pp. 126–155 (2017)
11. Komano, Y., Mizuki, T.: Physical zero-knowledge proof protocol for topswops. In: Proc. of 17th International Conference on Information Security Practice and Experience (ISPEC 2022), LNCS. Springer (2022)
12. Lafourcade, P., Miyahara, D., Mizuki, T., Robert, L., Sasaki, T., Sone, H.: How to construct physical zero-knowledge proofs for puzzles with a "single loop" condition. Theoretical Computer Science **888**, 41–55 (2021)
13. Manabe, Y.: Survey: Card-based cryptographic protocols to calculate primitives of boolean functions. International Journal of Computer & Software Engineering **27**(1), 178 (2022)
14. Miyahara, D., Hayashi, Y.i., Mizuki, T., Sone, H.: Practical card-based implementations of yao's millionaire protocol. Theoretical Computer Science **803**, 207–221 (2020)
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: Proc. of 10th International Conference on Fun with Algorithms (FUN 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
16. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for kakuro. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **102**(9), 1072–1078 (2019)
17. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Proc. of 12th International Conference on Unconventional Computing and Natural Computation (UCNC 2013), LNCS Vol. 7956. pp. 162–173 (2013)
18. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security **13**(1), 15–23 (2014)
19. Mizuki, T., Sone, H.: Six-card secure and and four-card secure xor. In: Proc. of 3rd International Workshop on Frontiers in Algorithms(FAW 2009), LNCS Vol. 5598. pp. 358–369 (2009)
20. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: How to solve millionaires' problem with two kinds of cards. New Generation Computing **39**(1), 73–96 (2021)
21. Ono, H., Manabe, Y.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: Proc. of 13th Asia Joint Conference on Information Security(AsiaJCIS 2018). pp. 23–28 (2018)
22. Robert, L., Miyahara, D., Lafourcade, P., Libralesso, L., Mizuki, T.: Physical zero-knowledge proof and np-completeness proof of suguru puzzle. Information and Computation **285**, 104858 (2022)

23. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Card-based zkp for connectivity: Applications to nurikabe, hitori, and heyawake. New Generation Computing **40**(1), 149–171 (2022)
24. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Card-based zkp protocol for nurimisaki. In: Proc. of 24th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2022), LNCS. Springer (2022)
25. Ruangwises, S.: An improved physical zkp for nonogram. arXiv preprint arXiv:2106.14020 (2021)
26. Ruangwises, S.: Two standard decks of playing cards are sufficient for a zkp for sudoku. New Generation Computing **40**(1), 49–65 (2022)
27. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for numberlink puzzle and k vertex-disjoint paths problem. New Generation Computing **39**(1), 3–17 (2021)
28. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for ripple effect. Theoretical Computer Science **895**, 115–123 (2021)
29. Ruangwises, S., Itoh, T.: Physical zkp for makaro using a standard deck of cards. arXiv preprint arXiv:2112.12042 (2021)
30. Ruangwises, S., Itoh, T.: How to physically verify a rectangle in a grid: A physical zkp for shikaku. arXiv preprint arXiv:2202.09788 (2022)
31. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for sudoku. Theoretical Computer Science **839**, 135–142 (2020)
32. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any boolean circuit. Discrete Applied Mathematics **289**, 248–261 (2021)