Physical Zero-Knowledge Proof for Ripple Effect

Suthee Ruangwises(✉) and Toshiya Itoh

Department of Mathematical and Computing Science,
Tokyo Institute of Technology, Tokyo, Japan
ruangwises@gmail.com, titoh@c.titech.ac.jp

Abstract. Ripple Effect is a logic puzzle with an objective to fill numbers into a rectangular grid divided into rooms. Each room must contain consecutive integers starting from 1 to its size. Also, if two cells in the same row or column have the same number x, the space separating the two cells must be at least x cells. In this paper, we propose a physical protocol of zero-knowledge proof for Ripple Effect puzzle using a deck of cards, which allows a prover to physically show that he/she knows a solution without revealing it. In particular, we develop a physical protocol that, given a secret number x and a list of numbers, verifies that x does not appear among the first x numbers in the list without revealing x or any number in the list.

Keywords: Zero-knowledge proof · Card-based cryptography · Ripple effect · Puzzle

1 Introduction

Ripple Effect is a logic puzzle introduced by Nikoli, a Japanese company that developed many famous logic puzzles such as Sudoku, Numberlink, and Kakuro. A Ripple Effect puzzle consists of a rectangular grid of size $m \times n$ divided into polyominoes called *rooms*, with some cells already containing a number (we call these cells *fixed cells* and the other cells *empty cells*). The objective of this puzzle is to fill a number into each empty cell according to the following rules [13].

1. *Room condition*: Each room must contain consecutive integers starting from 1 to its *size* (the number of cells in the room).
2. *Distance condition*: If two cells in the same row or column have the same number x, the space separating the two cells must be at least x cells. See Fig. 1.

Suppose that Patricia, a Ripple Effect expert, created a difficult Ripple Effect puzzle and challenged her friend Victor to solve it. After a while, Victor could not solve her puzzle and began to doubt that the puzzle may have no solution. Patricia needs to convince him that her puzzle actually has a solution without showing it (which would make the challenge pointless). In this situation, Patricia needs a protocol of *zero-knowledge proof*.

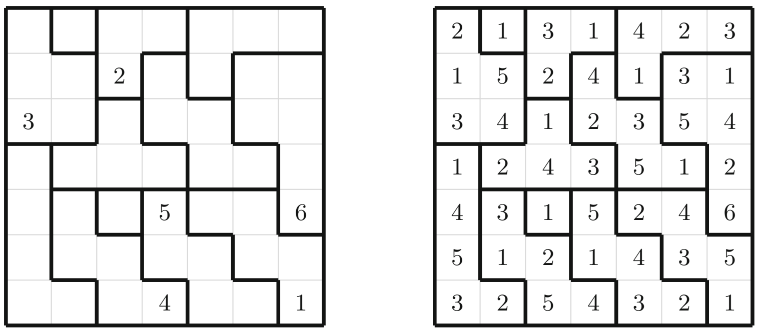

Fig. 1. An example of a Ripple Effect puzzle (left) and its solution (right)

1.1 Zero-Knowledge Proof

A zero-knowledge proof is a protocol of interactive proof between a prover P and a verifier V. Both P and V are given a computational problem x, but only P knows a solution w of x. A protocol of zero-knowledge proof enables P to convince V that he/she knows w without revealing any information of it. The protocol must satisfy the following properties.

1. **Completeness:** If P knows w, then P can convince V with high probability. (In this paper, we consider only the *perfect completeness* property where the probability to convince V is one.)
2. **Soundness:** If P does not know w, then P cannot convince V, except with a small probability called *soundness error*. (In this paper, we consider only the *perfect soundness* property where the soundness error is zero.)
3. **Zero-knowledge:** V learns nothing about w, i.e. there exists a probabilistic polynomial time algorithm S (called a *simulator*), not knowing w but having a black-box access to V, such that the outputs of S and the outputs of the real protocol follow the same probability distribution.

The concept of a zero-knowledge proof was introduced by Goldwasser et al. [6] in 1989. Goldreich et al. [5] later showed that a zero-knowledge proof exists for every NP problem. As Ripple Effect has been proved to be NP-complete [17], one can construct a computational zero-knowledge proof for it. However, such construction is not intuitive or practical as it requires cryptographic primitives.

Instead, we aim to develop a physical protocol of zero-knowledge proof with a deck of playing cards. Card-based protocols have benefit that they do not require computers and use only a small, portable deck of cards that can be found in everyday life. These protocols are also suitable for teaching purpose since they are easy to understand and verify the correctness and security, even for non-experts in cryptography.

1.2 Related Work

Development of card-based protocols of zero-knowledge proof for logic puzzles began with a protocol for Sudoku developed by Gradwohl et al. [7]. However, each of several variants of this protocol either uses scratch-off cards or has a nonzero soundness error. Later, Sasaki et al. [15] improved the protocol for Sudoku to achieve perfect soundness property without using special cards. Apart from Sudoku, protocols of zero-knowledge proof for other puzzles have been developed as well, including Nonogram [3], Akari [1], Takuzu [1], Kakuro [1,12], KenKen [1], Makaro [2], Norinori [4], Slitherlink [11], and Numberlink [14].

Many of these protocols employ methods to physically verify or compute specific number-related functions, as shown in the following examples.

- A subprotocol in [7] verifies that a list is a permutation of all given numbers in some order without revealing their order.
- A subprotocol in [2] verifies that two given numbers are different without revealing their values.
- Another subprotocol in [2] verifies that a number in a list is the largest one in that list without revealing any value in the list.
- A subprotocol in [14] counts the number of elements in a list that are equal to a given secret value without revealing that value, the positions of elements in the list that are equal to it, or the value of any other element in the list.

Observe that these four functions do not use the mathematical meaning of numbers. In these functions, numbers are treated only as symbols distinguished from one another in the sense that we can replace every number x with $f(x)$ for any function $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ and the output will remain the same (for the third example, f has to be an increasing function).[1]

1.3 Our Contribution

In this paper, we propose a physical protocol of zero-knowledge proof with perfect completeness and perfect soundness properties for Ripple Effect puzzle using a deck of cards. More importantly, we extend the set of functions that are known to be physically verifiable. In particular, we develop a protocol that, given a secret number x and a list of numbers, verifies that x does not appear among the first x numbers in the list without revealing x or any number in the list.

Unlike the functions verified by many protocols in previous work, the function our protocol has to verify uses the mathematical meaning of the numbers in the sense of cardinality; it uses the value of x to determine how many elements in the list the condition is imposed on. Therefore, this function is significantly harder to verify *without revealing* x, and thus we consider this result to be an important step in developing protocols of zero-knowledge proof.

[1] Actually, some protocols from previous work can verify a function that uses the mathematical meaning of numbers, but still not in the sense of cardinality. For example, a subprotocol in [12] verifies that the sum of all numbers in a list is equal to a given number; for this function, we can still replace every number x with $f(x)$ for any linear function $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$.

2 Preliminaries

2.1 Cards

Every card used in our protocol has either $\clubsuit$ or $\heartsuit$ on the front side, and has an identical back side.

For $1 \leq x \leq y$, define $E_y(x)$ to be a sequence of consecutive y cards arranged horizontally, with all of them being $\boxed{\clubsuit}$ except the x-th card from the left being $\boxed{\heartsuit}$, e.g. $E_3(3)$ is $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}$ and $E_4(2)$ is $\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$. We use $E_y(x)$ to encode a number x in a situation where the maximum possible number is at most y. This encoding rule was first considered by Shinagawa et al. [16] in the context of using a regular y-gon card to encode each integer in $\mathbb{Z}/y\mathbb{Z}$. Additionally, we encode 0 by $E_y(0)$, defined to be a sequence of consecutive y cards, all of them being $\boxed{\clubsuit}$.

Normally, the cards in $E_y(x)$ are arranged horizontally as defined above unless stated otherwise. However, in some situations we may arrange the cards vertically, where the leftmost card will become the topmost card and the rightmost card will become the bottommost card (hence the only $\boxed{\heartsuit}$ will become the x-th topmost card for $x \geq 1$).

2.2 Matrix

We construct an $a \times b$ *matrix* of face-down cards. Let Row i denote the i-th topmost row, and Column j denote the j-th leftmost column. Let $M(i, j)$ denote the card at Row i and Column j of a matrix M. See Fig. 2.

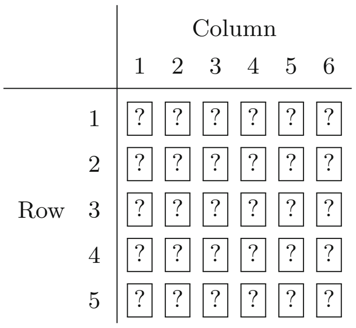

Fig. 2. An example of a 5×6 matrix

2.3 Pile-Shifting Shuffle

A *pile-shifting shuffle* was introduced by Shinagawa et al. [16]. In the pile-shifting shuffle, we rearrange the columns of the matrix by a random cyclic permutation,

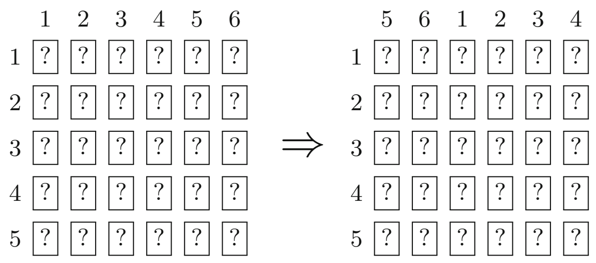

Fig. 3. An example of a pile-shifting shuffle on a 5 $\times$ 6 matrix with $r = 2$

i.e. move every Column ℓ to Column $\ell + r$ for a uniformly random $r \in \{0, 1, ..., b-1\}$ (where Column ℓ' means Column $\ell' - b$ for $\ell' > b$). See Fig. 3.

One can perform the pile-shifting shuffle in real world by putting the cards in each column into an envelope and then applying a *Hindu cut*, which is a basic shuffling operation commonly used in card games [18], to the sequence of envelopes.

2.4 Rearrangement Protocol

The sole purpose of a *rearrangement protocol* is to revert columns of a matrix (after we perform pile-shifting shuffles) back to their original positions so that we can reuse all cards in the matrix without revealing them. Slightly different variants of this protocol were used in some previous work on card-based protocols [2,8,9,14,15]. Note that throughout our main protocol, we always put $E_b(1)$ in Row 1 when constructing a new matrix, hence we want to ensure that a $\boxed{\heartsuit}$ in Row 1 moves back to Column 1.

We can apply the rearrangement protocol on an $a \times b$ matrix by publicly performing the following steps.

1. Apply the pile-shifting shuffle to the matrix.
2. Turn over all cards in Row 1. Locate the position of a $\boxed{\heartsuit}$. Suppose it is at Column j. Turn over all face-up cards.
3. Shift the columns of the matrix to the left by $j - 1$ columns, i.e. move every Column ℓ to Column $\ell - (j - 1)$ (where Column ℓ' means Column $\ell' + b$ for $\ell' < 1$).

2.5 Uniqueness Verification Protocol

Suppose we have sequences $S_0, S_1, ..., S_a$, each consisting of b cards. S_0 encodes a positive number, while $S_1, S_2, ..., S_a$ encode nonnegative numbers. Our objective is to verify that none of the sequences $S_1, S_2, ..., S_a$ encodes the same number as S_0 without revealing any number. This protocol is a special case of the protocol developed by Ruangwises and Itoh [14] to count the number of indices i such that S_i encodes the same number as S_0. We can do so by publicly performing the following steps.

1. Construct an $(a+2) \times b$ matrix with Row 1 consisting of a sequence $E_b(1)$ and each Row $i+2$ $(i = 0, 1, ..., a)$ consisting of the sequence S_i.
2. Apply the pile-shifting shuffle to the matrix.
3. Turn over all cards in Row 2. Locate the position of a $\boxed{\heartsuit}$. Suppose it is at Column j.
4. Turn over all cards in Column j from Row 3 to Row $a+2$. If there is no $\boxed{\heartsuit}$ among them, then the protocol continues. Otherwise, V rejects and the protocol terminates.
5. Turn over all face-up cards.

2.6 Pile-Scramble Shuffle

A *pile-scramble shuffle* was introduced by Ishikawa et al. [10]. In the pile-scramble shuffle, we rearrange the columns of the matrix by a random permutation, i.e. move every Column j to Column p_j for a uniformly random permutation $p = (p_1, p_2, ..., p_b)$ of $(1, 2, ..., b)$. See Fig. 4.

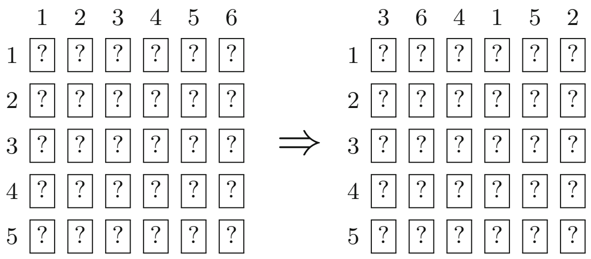

Fig. 4. An example of a pile-scramble shuffle on a 5×6 matrix

One can perform the pile-scramble shuffle in real world by putting the cards in each column into an envelope and then scrambling the envelopes together completely randomly.

3 Main Protocol

Let k be the size of the biggest room in the Ripple Effect grid. For each fixed cell with a number x, the prover P publicly puts a sequence of face-down cards $E_k(x)$ on it. Then, for each empty cell with a number x in P's solution, P secretly puts a sequence of face-down cards $E_k(x)$ on it. P will first verify the distance condition, and then the room condition.

3.1 Verification Phase for Distance Condition

The most challenging part of our protocol is to verify the distance condition, which is equivalent to the following statement: for each cell c with a number x, the first x cells to the right and to the bottom of c cannot have a number x.

First, we will show a protocol to verify that there is no number x among the first x cells to the right of c. Then, we apply the same protocol analogously in the direction to the bottom of c as well.

Suppose that cell c has a number x. Let A_0 be the sequence of cards on c. For each $i = 1, 2, ..., k$, let A_i be the sequence of cards on the i-th cell to the right of c, i.e. A_1 is on a cell right next to the right of c, A_2 is on a second cell to the right of c, and so on (if there are only $\ell < k$ cells to the right of c, we publicly put $E_k(0)$ in place of A_i for every $i > \ell$).

The intuition of this protocol is that, we will create sequences $B_1, B_2, ..., B_{k-1}$, all being $E_k(0)$, and insert them between A_x and A_{x+1} (without revealing x). Then, we will pick k sequences $A_1, A_2, ..., A_x, B_1, B_2, ..., B_{k-x}$ and use the uniqueness verification protocol introduced in Sect. 2.5 to verify that none of them encodes the same number as A_0.

P publicly performs the following steps.

1. Construct a $(k + 4) \times k$ matrix M by the following procedures. See Fig. 5.
   – In Row 1, place a sequence $E_k(1)$.
   – In Row 2, place the sequence A_0 (which is $E_k(x)$).
   – In Row 3, place a sequence $E_k(1)$.
   – In Row 4, place a sequence $E_k(0)$.
   – In each Column j ($j = 1, 2, ..., k$), place the sequence A_j arranged vertically from Row 5 to Row $k + 4$.
2. Apply the pile-shifting shuffle to M.
3. Turn over all cards in Row 2 of M. Locate the position of a $\boxed{\heartsuit}$. Suppose it is at Column j_1. Turn over all face-up cards.
4. Shift the columns of M to the right by $k - j_1$ columns, i.e. move every Column ℓ to Column $\ell + k - j_1$ (where Column ℓ' means Column $\ell' - k$ for $\ell' > k$). Observe that after this step, A_x will locate at the rightmost column.
5. Divide M into a $2 \times k$ matrix M_1 and a $(k + 2) \times k$ matrix M_2. M_1 consists of the topmost two rows of M, while M_2 consists of everything below M_1. Each cell $M(i + 2, j)$ $(i, j \geq 1)$ of M will become a cell $M_2(i, j)$ of a new matrix M_2.
6. Apply the rearrangement protocol to M_1. Observe that we now have $E_k(1)$ in Row 1 and A_0 in Row 2 of M_1. From now on, we will perform operations only on M_2 while M_1 will be left unchanged.
7. Append $k - 1$ columns to the right of the matrix M_2 by the following procedures, making M_2 become a $(k + 2) \times (2k - 1)$ matrix.
   – In Row 1, place a sequence $E_{k-1}(0)$ from Column $k+1$ to Column $2k-1$.
   – In Row 2, place a sequence $E_{k-1}(1)$ from Column $k+1$ to Column $2k-1$.
   – In each Column $k+j$ ($j = 1, 2, ..., k-1$), place a sequence $E_k(0)$ arranged vertically from Row 3 to Row $k + 2$. We call this sequence B_j.

8. Apply the pile-shifting shuffle to M_2.
9. Turn over all cards in Row 1 of M_2. Locate the position of a $\boxed{\heartsuit}$. Suppose it is at Column j_2. Turn over all face-up cards.
10. For each $i = 1, 2, ..., k$, let S_i denote a sequence of card arranged vertically at Column $j_2 + i - 1$ (where Column ℓ' means Column $\ell' - (2k - 1)$ for $\ell' > 2k - 1$) from Row 3 to Row $k + 2$ of M_2. Observe that $(S_1, S_2, ..., S_k) = (A_1, A_2, ..., A_x, B_1, B_2, ..., B_{k-x})$. Then, construct a $(k + 2) \times k$ matrix N with Row 1 consisting of a sequence $E_k(1)$ taken from Row 1 of M_1, Row 2 consisting of the sequence A_0 taken from Row 2 of M_1, and each Row $i + 2$ $(i = 1, 2, ..., k)$ consisting of the sequence S_i taken from M_2.
11. Apply the uniqueness verification protocol on N. The intuition of this step is to verify that none of the sequences $A_1, A_2, ..., A_x$ encodes the same number as A_0 (while $B_1, B_2, ..., B_{k-x}$ are all $E_k(0)$).
12. Apply the rearrangement protocol on N, put A_0 back onto c, and put $S_1, S_2, ..., S_k$ back to their corresponding columns in M_2.
13. Apply the pile-shifting shuffle to M_2.
14. Turn over all cards in Row 2 of M_2. Locate the position of a $\boxed{\heartsuit}$. Suppose it is at Column j_3. Turn over all face-up cards.
15. Shift the columns of M_2 to the right by $k + 1 - j_3$ columns, i.e. move every Column ℓ to Column $\ell+k+1-j_3$ (where Column ℓ' means Column $\ell'-(2k-1)$ for $\ell' > 2k-1$). Then, remove Columns $k+1, k+2, ..., 2k-1$ from M_2, making M_2 become a $(k + 2) \times k$ matrix again. Observe that the columns we just removed are exactly the same $k - 1$ columns we previously appended to M_2.
16. Apply the rearrangement protocol on M_2 and put the sequences $A_1, A_2, ..., A_k$ back onto their corresponding cells on the Ripple Effect grid.

P performs these steps analogously in the direction to the right and bottom of every cell in the grid. If every cell passes the verification, P continues to the verification phase for room condition.

3.2 Verification Phase for Room Condition

The room condition of Ripple Effect is exactly the same as that of Makaro, and hence can be verified by a subprotocol in [2]. Since this is the final step of our protocol, after we finish verifying each room, we do not have to rearrange cards back to their original positions or put them back onto their cells.

P will verify each room separately. For a room R with size s, let $A_1, A_2, ..., A_s$ be the sequences of cards on the cells in R in any order. To verify room R, P publicly performs the following steps.

1. Construct a $k \times s$ matrix M by the following procedures: in each Column j $(j = 1, 2, ..., s)$, place the sequence A_j arranged vertically from Row 1 to Row k.
2. Apply the pile-scramble shuffle to M.
3. Turn over all cards in M. If all columns of M are a permutation of $E_k(1)$, $E_k(2), ..., E_k(s)$ arranged vertically, then the protocol continues. Otherwise, V rejects and the protocol terminates.

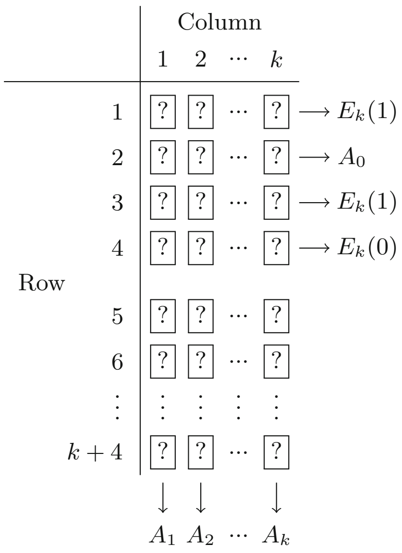

Fig. 5. A $(k+4) \times k$ matrix M constructed in Step 1

P performs these steps for every room. If every room passes the verification, then V accepts.

In total, our protocol uses $kmn + 2k^2 + 4k - 2 = \Theta(kmn)$ cards.

4 Proof of Security

We will prove the perfect completeness, perfect soundness, and zero-knowledge properties of our protocol. We omit the proofs of the verification phase for room condition as they have been shown in [2].

Lemma 1 (Perfect Completeness). *If P knows a solution of the Ripple Effect puzzle, then V always accepts.*

Proof. First, we will show that the uniqueness verification protocol will pass if none of $S_1, S_2, ..., S_a$ encodes the same number as S_0. Suppose that S_0 encodes a number $z > 0$. A $\boxed{\heartsuit}$ in Row 2 will locate at Column z. Since none of $S_1, S_2, ..., S_a$ encodes the number z, all cards below Row 2 in the same column as that $\boxed{\heartsuit}$ will be all $\boxed{\clubsuit}$s. This remains true after we rearrange the columns in Step 2. Therefore, the verification in Step 4 will pass.

Now consider the main protocol. Suppose that P knows a solution of the puzzle. The verification phase for room condition will pass [2].

Consider the verification phase for distance condition. In Step 1, a $\boxed{\heartsuit}$ in Row 2 is at the same column as A_x, and will always be. Therefore, in Step 4, the column containing A_x will move to the rightmost column.

In Step 7, the order of the sequences (which are arranged vertically from Row 3 to Row $k+2$) from the leftmost column to the rightmost column is $A_{x+1}, A_{x+2}, ..., A_k, A_1, A_2, ..., A_x, B_1, B_2, ..., B_{k-1}$. Also, a $\boxed{\heartsuit}$ in Row 1 is at the same column as A_1, and a $\boxed{\heartsuit}$ in Row 3 is at the same column as B_1, and they will always be.

In Step 10, since A_1 locates at Column j, the sequences $S_1, S_2, ..., S_k$ will be exactly $A_1, A_2, ..., A_x, B_1, B_2, ..., B_{k-x}$ in this order. Therefore, in Step 11, the uniqueness verification protocol will pass because none of the sequences $A_1, A_2, ..., A_x$ encodes the number x, and $B_1, B_2, ..., B_{k-x}$ all encode 0.

Since this is true for every cell and every direction (to the right, left, top, and bottom), the verification phase for distance condition will pass, hence V will always accept. □

Lemma 2 (Perfect Soundness). *If P does not know a solution of the Ripple Effect puzzle, then V always rejects.*

Proof. First, we will show that the uniqueness verification protocol will fail if at least one of $S_1, S_2, ..., S_a$ encodes the same number as S_0. Suppose that S_0 and S_d ($d > 0$) both encode a number $z > 0$. A $\boxed{\heartsuit}$ in Row 2 will locate at Column z. Since S_d also encodes the number z, a card in Row d in the same column as that $\boxed{\heartsuit}$ will be a $\boxed{\heartsuit}$. This remains true after we rearrange the columns in Step 2. Therefore, the verification in Step 4 will fail.

Now consider the main protocol. Suppose that P does not know a solution of the puzzle. The numbers P puts into the grid must violate either the room condition or the distance condition. If they violate the room condition, the verification phase for room condition will fail [2].

Suppose that the numbers in the grid violate the distance condition. There must be two cells c and c' in the same row or column having the same number x, where c' locates on the right or the bottom of c with $\ell < x$ cells of space between them.

Consider when P performs the verification phase for distance condition for c in the direction towards c'. The sequence on the cell c' will be $A_{\ell+1}$

By the same reason as in the proof of Lemma 1, in Step 10, the sequences $S_1, S_2, ..., S_k$ will be exactly $A_1, A_2, ..., A_x, B_1, B_2, ..., B_{k-x}$ in this order and thus include $A_{\ell+1}$. Therefore, in Step 11, the uniqueness verification protocol will fail because A_ℓ encodes the number x, hence V will always reject. □

Lemma 3 (Zero-Knowledge). *During the verification phase, V learns nothing about P's solution of the Ripple Effect puzzle.*

Proof. To prove the zero-knowledge property, it is sufficient to prove that all distributions of the values that appear when P turns over cards can be simulated by a simulator S without knowing P's solution.

- In the rearrangement protocol:
  - Consider Step 2 where we turn over all cards in Row 1. This occurs right after we applied the pile-shifting shuffle to the matrix. Therefore,

a $\boxed{\heartsuit}$ has an equal probability to appear at each of the b columns, hence this step can be simulated by S without knowing P's solution.

- In the uniqueness verification protocol:
  - Consider Step 3 where we turn over all cards in Row 2. This occurs right after we applied the pile-shifting shuffle to the matrix. Therefore, a $\boxed{\heartsuit}$ has an equal probability to appear at each of the b columns, hence this step can be simulated by S without knowing P's solution.
  - Consider Step 4 where we turn over all cards in Column j from Row 3 to Row $a+2$. If the verification passes, the cards we turn over must be all $\boxed{\clubsuit}$s, hence this step can be simulated by S without knowing P's solution.
- In the verification phase for room condition:
  - Consider Step 3 where we turn over all cards in Row 2 of M. This occurs right after we applied the pile-shifting shuffle to M. Therefore, a $\boxed{\heartsuit}$ has an equal probability to appear at each of the k columns, hence this step can be simulated by S without knowing P's solution.
  - Consider Step 9 where we turn over all cards in Row 1 of M_2. This occurs right after we applied the pile-shifting shuffle to M_2. Therefore, a $\boxed{\heartsuit}$ has an equal probability to appear at each of the $2k-1$ columns, hence this step can be simulated by S without knowing P's solution.
  - Consider Step 14 where we turn over all cards in Row 2 of M_2. This occurs right after we applied the pile-shifting shuffle to M_2. Therefore, a $\boxed{\heartsuit}$ has an equal probability to appear at each of the $2k-1$ columns, hence this step can be simulated by S without knowing P's solution.

Therefore, we can conclude that V learns nothing about P's solution. □

5 Future Work

We developed a physical protocol of zero-knowledge proof for Ripple Effect puzzle using $\Theta(kmn)$ cards. A possible future work is to develop a protocol for this puzzle that requires asymptotically fewer number of cards, or for other popular logic puzzles.

Another challenging future work is to explore methods to physically verify other types of more complicated number-related functions.

References

1. Bultel, X., Dreier, J., Dumas, J.-G., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: Proceedings of the 8th International Conference on Fun with Algorithms (FUN), pp. 8:1–8:20 (2016)
2. Bultel, X., et al.: Physical zero-knowledge proof for Makaro. In: Proceedings of the 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), pp. 111–125 (2018)
3. Chien, Y.-F., Hon, W.-K.: Cryptographic and physical zero-knowledge proof: from Sudoku to Nonogram. In: Proceedings of the 5th International Conference on Fun with Algorithms (FUN), pp. 102–112 (2010)

4. Dumas, J.-G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: Proceedings of the 25th International Computing and Combinatorics Conference (COCOON), pp. 166–177 (2019)
5. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. J. ACM **38**(3), 691–729 (1991)
6. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
7. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. In: Proceedings of the 4th International Conference on Fun with Algorithms (FUN), pp. 166–182 (2007)
8. Hashimoto, Y., Shinagawa, K., Nuida, K., Inamura, M., Hanaoka, G.: Secure grouping protocol using a deck of cards. In: Proceedings of the 10th International Conference on Information Theoretic Security (ICITS), pp. 135–152 (2017)
9. Ibaraki, T., Manabe, Y.: A more efficient card-based protocol for generating a random permutation without fixed points. In: Proceedings of the 3rd International Conference on Mathematics and Computers in Sciences and Industry (MCSI), pp. 252–257 (2016)
10. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Proceedings of the 14th International Conference on Unconventional Computation and Natural Computation (UCNC), pp. 215–226 (2015)
11. Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: A physical ZKP for Slitherlink: how to perform physical topology-preserving computation. In: Proceedings of the 15th International Conference on Information Security Practice and Experience (ISPEC), pp. 135–151 (2019)
12. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for Kakuro. IEICE Trans. Fundamentals Electron. Commun. Comput. Sci. **E102.A**(9), 1072–1078 (2019)
13. Nikoli: Ripple Effect. https://www.nikoli.co.jp/en/puzzles/ripple_effect.html
14. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for numberlink. In: Proceedings of the 10th International Conference on Fun with Algorithms (FUN), pp. 22:1–22:11 (2020)
15. Sasaki, T., Mizuki, T., Sone, H.: Card-based zero-knowledge proof for Sudoku. In: Proceedings of the 9th International Conference on Fun with Algorithms (FUN), pp. 29:1–29:10 (2018)
16. Shinagawa, K., et al.: Multi-party computation with small shuffle complexity using regular polygon cards. In: Proceedings of the 9th International Conference on Provable Security (ProvSec), pp. 127–146 (2015)
17. Takenaga, Y., Aoyagi, S., Iwata, S., Kasai, T.: Shikaku and ripple effect are NP-complete. Congressus Numerantium **216**, 119–127 (2013)
18. Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: How to implement a random bisection cut. In: Proceedings of the 5th International Conference on the Theory and Practice of Natural Computing (TPNC), pp. 58–69 (2016)