Physical Zero-Knowledge Proof for Numberlink

Suthee Ruangwises
Department of Mathematical and Computing Science, Tokyo Institute of Technology, Japan
ruangwises.s.aa@m.titech.ac.jp

Toshiya Itoh
Department of Mathematical and Computing Science, Tokyo Institute of Technology, Japan
titoh@c.titech.ac.jp

Abstract

Numberlink is a logic puzzle for which the player has to connect all pairs of cells with the same numbers by non-crossing paths in a rectangular grid. In this paper, we propose a physical protocol of zero-knowledge proof for Numberlink using a deck of cards, which allows a player to physically show that he/she knows a solution without revealing it. In particular, we develop a physical protocol to count 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. Our protocol can also be applied to verify the existence of vertex-disjoint paths connecting all given pairs of endpoints in any undirected graph.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Theory of computation → Cryptographic protocols

Keywords and phrases Zero-knowledge proof, Card-based cryptography, Numberlink, Puzzles, Games

Digital Object Identifier 10.4230/LIPIcs.FUN.2021.22

1 Introduction

Numberlink is a logic puzzle introduced by a Japanese company Nikoli famous for developing many popular puzzles including Sudoku, Akari, Makaro, and Norinori. The puzzle has become increasingly popular and a large number of Numberlink mobile apps with different names and slightly different variants of rule have been developed [8].

A Numberlink puzzle consists of a rectangular grid with some cells containing a number. Each number appears exactly twice in the grid. The goal of this puzzle is to connect every pair of the same numbers by a path that can go from a cell to its horizontally or vertically adjacent cell. Paths cannot cross or share a cell with one another. In the official rule [16], it is not required that all cells in the grid have to be covered by paths. However, a puzzle is generally considered to be well-designed if it has a unique solution, and all cells are covered by paths in that solution.

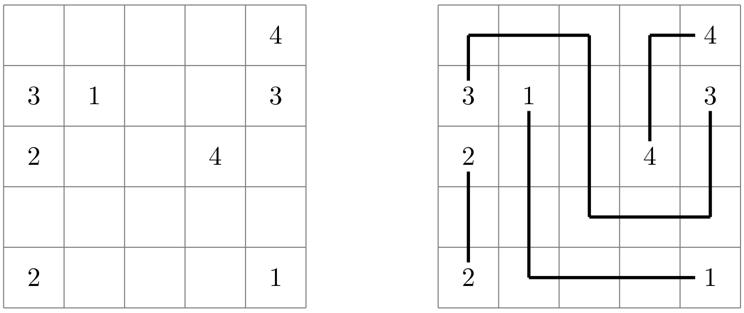


Figure 1 An example of a Numberlink puzzle (left) and its solution (right).

10th International Conference on Fun with Algorithms (FUN 2021).
Editors: Martin Farach-Colton, Giuseppe Prencipe, and Ruyhei Uehara; Article No. 22; pp. 22:1–22:11
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Suppose that Anna, an expert in Numberlink, created a difficult Numberlink puzzle and challenged her friend Brian to solve it. Unluckily, after several tries Brian could not solve her puzzle. He then claimed that the puzzle has no solution and refused to try it anymore. How can Anna convince bad-luck Brian that her puzzle actually has a solution without revealing it to him (which would render the challenge pointless)?

1.1 Zero-Knowledge Proof

A zero-knowledge proof is an interactive proof between a prover P and a verifier V, with both given an instance x of a computational problem. Only P knows a solution w of x, and the computational power of V is limited so that he/she cannot obtain w from x. P wants to convince V that he/she knows w without revealing any information about w to V. A zero-knowledge proof must satisfies the following three properties.

1. **Completeness:** If P knows w, then P is able to convince V with high probability (in this paper, we are interested in the *perfect completeness* property where the probability to convince V is one).
2. **Soundness:** If P does not know w, then P is not able to convince V, except with a small probability called *soundness error* (in this paper, we are interested in the *perfect soundness* property where the soundness error is zero).
3. **Zero-Knowledge:** V cannot obtain any information about w, i.e. there exists a probabilistic polynomial time algorithm S (called the simulator) that does not know w, and the outputs of S follow the same probability distribution as the outputs of the real protocol.

The concept of zero-knowledge proof was first introduced by Goldwasser et al. [7], and it was proved by Goldreich et al. [6] that a zero-knowledge proof exists for any NP problem. Because Numberlink has been proved to be NP-complete [1, 13, 14], it is possible to construct a cryptographic zero-knowledge proof for Numberlink. However, such construction requires cryptographic primitives and is not intuitive or practical.

Instead, we are interested in constructing a physical protocol of zero-knowledge proof using a deck of playing cards. The benefit of such protocols is that they use only a small deck of cards which can be found in everyday life and do not require computers. Moreover, these intuitive protocols are easy to understand and verify the security and correctness, even for non-experts, and thus can be used as examples for didactic purpose.

1.2 Related Work

In 2007, Gradwohl et al. [9] developed the first physical protocols of zero-knowledge proof for Sudoku. Each of their several variants of the protocol either uses special scratch-off cards or has a non-zero soundness error. Later, Sasaki et al. [17] improved the protocol for Sudoku to achieve perfect soundness without using special cards. Besides Sudoku, physical protocols of zero-knowledge proof for other logic puzzles have been developed as well, including Nonogram [4], Akari [2], Takuzu [2], Kakuro [2, 15], KenKen [2], Makaro [3], and Norinori [5].

These protocols of zero-knowledge proof employ methods to physically verify specific functions. For example, the protocol for Sudoku [9] shows how to verify the presence of all numbers in a list without revealing their order, the protocol for Makaro [3] shows how to verify that a number is the largest one in a list without revealing any value in the list, and the protocol for Norinori [5] shows how to verify the presence of a given number in a list without revealing its position or any other value in the list.

1.3 Our Contribution

In this paper, we propose a physical protocol of zero-knowledge proof with perfect completeness and perfect soundness for Numberlink using a deck of cards. More importantly, we also extend the set of functions that are known to be physically verifiable.

By developing the protocol for Numberlink, we show in particular how to count 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. Also, by transforming a graph problem into this element-counting problem, we show how to verify the existence of vertex-disjoint paths connecting all given pairs of endpoints in any undirected graph.

2 Preliminaries

2.1 Numberlink Board

Suppose that a Numberlink grid has size $m \times n$, and has k pairs of numbers $1, 2, ..., k$ written on it. We call two cells in the grid *adjacent* if they are horizontally or vertically adjacent. Cells with a number written on them are called *terminal cells*; other cells are called *non-terminal cells*.

A *path* in a valid solution of a Numberlink puzzle is a sequence of cells $(c_1, c_2, ..., c_t)$ where c_1 and c_t are terminal cells with the same numbers written on them and all other cells are non-terminal cells, with c_i being adjacent to c_{i+1} for every $i = 1, 2, ..., t-1$. Also, a path $(c_1, c_2, ..., c_t)$ is called *simple* if there is no i, j such that $j > i+1$ and c_i is adjacent to c_j.

A Numberlink puzzle is called *well-designed* if it has a unique solution, and all cells are covered by paths in that solution. Observe that if a puzzle is well-designed, then every path in its solution must be simple (otherwise if we have a non-simple path $(c_1, c_2, ..., c_t)$ with c_i being adjacent to c_j where $j > i+1$, then we can replace it with a shorter path $(c_1, c_2, ...c_i, c_j, c_{j+1} ..., c_t)$, thus creating an alternative solution).

2.2 Cards

In our protocol, we use two types of cards: *encoding cards* and *marking cards*. An encoding card has either ♣ or ♡ on the front side, while a marking card has a positive integer on the front side. All cards have an identical back side.

Define $E_y(x)$ to be a sequence of y encoding cards, with all cards being $\boxed{\clubsuit}$ except the x-th card from the left being $\boxed{\heartsuit}$. For example, $E_3(1)$ is $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$ and $E_4(3)$ is $\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$. We use $E_y(x)$ to encode a number x in the situation where the maximum possible number is at most y.

2.3 Matrix

Suppose we have a numbers $x_1, x_2, ..., x_a$, with each of them being at most b. Each number x_i is encoded by a sequence of cards $E_b(x_i)$. We construct a matrix $D(a, b)$ of cards by the following procedures.

First, create an $a \times b$ matrix of face-down encoding cards, with the i-th topmost row being $E_b(x_i)$. Then, on top of the topmost row of the matrix, place face-down marking cards $1, 2, ..., b$ from left to right in this order. We call this new row Row 0. Also, to the left of the leftmost column of the matrix, place face-down marking cards $2, 3, ..., a$ from top to bottom

$$\begin{array}{cc|ccccccccl}
 & & & & & \text{Column} & & & & & \\
 & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & & \\
\hline
 & 0 & & \boxed{1} & \boxed{2} & \boxed{3} & \boxed{4} & \boxed{5} & \boxed{6} & & \text{(actually face-down)} \\
 & 1 & & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \longrightarrow & E_6(x_1) \\
 & 2 & \boxed{2} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \longrightarrow & E_6(x_2) \\
\text{Row} & 3 & \boxed{3} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \longrightarrow & E_6(x_3) \\
 & 4 & \boxed{4} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \longrightarrow & E_6(x_4) \\
 & 5 & \boxed{5} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \boxed{?} & \longrightarrow & E_6(x_5) \\
 & & \hookrightarrow & \text{(actually face-down)} & & & & & & &
\end{array}$$

Figure 2 An example of a matrix $D(5,6)$.

in this order (starting at Row 2). We call this new column Column 0. As a result, $D(a,b)$ becomes an incomplete $(a+1)\times(b+1)$ matrix with two cards at the top-left corner removed (see Figure 2).

We will then introduce the operations that will be applied to the matrix $D(a,b)$.

2.4 Double-Scramble Shuffle

A *double-scramble shuffle* is an extension of a *pile-scramble shuffle* first developed by Ishikawa et al. [12]. In the pile-scramble shuffle, we shuffle only the columns of the matrix by a random permutation; in the double-scramble shuffle, we shuffle both the selected rows and selected columns of the matrix by random permutations.

The formal procedures of the double-scramble shuffle are as follows.

1. Uniformly select a permutation $p = (p_2, p_3, ..., p_a)$ of $(2, 3, ..., a)$ at random. p must be unknown to the verifier.
2. Secretly rearrange Rows $2, 3, ..., a$ by a permutation p, i.e. move Row i to Row p_i for every $i = 2, 3, ..., a$.
3. Uniformly select a permutation $q = (q_1, q_2, ..., q_b)$ of $(1, 2, ..., b)$ at random. q must be unknown to the verifier.
4. Secretly rearrange Columns $1, 2, ..., b$ by a permutation q, i.e. move Column j to Column q_j for every $j = 1, 2, ..., b$.

Observe that the double-scramble shuffle makes the order of $x_2, x_3, ..., x_a$ indifferent to the verifier. It also hides the actual value of each x_i, but preserves the number of rows that encode the same value as Row 1.

▶ Remark 1. In real world, the prover P can perform the double-scramble shuffle by the following procedures. In Step 2, P publicly puts the cards in each row into an envelope and seal it. Then, P rearranges the envelopes by a permutation p without the verifier V observing. Finally, P publicly opens each envelope and put the cards in it back into a corresponding row. By doing this, V can ensure that P has only made row-wise swaps (and not arbitrary exchanges of cards) without knowing the permutation p. The same goes for column-wise swaps in Step 4.

2.5 Rearrangement Protocol

A rearrangement protocol was implicitly used in some previous work on card-based protocols [3, 10, 11, 17]. The sole purpose of this protocol is to revert the cards (after we perform some operations on them) back to their original positions so that we can reuse the cards without revealing them.

The formal procedures of the rearrangement protocol are as follows.

1. Apply the double-scramble shuffle to the matrix.
2. Publicly turn over all marking cards in Column 0. Suppose the opened cards are $p_2, p_3, ..., p_a$ from top to bottom in this order.
3. Publicly rearrange Rows $2, 3, ..., a$ by a permutation $p = (p_2, p_3, ..., p_a)$, i.e. move Row i to Row p_i for every $i = 2, 3, ..., a$.
4. Publicly turn over all marking cards in Row 0. Suppose the opened cards are $q_1, q_2, ..., q_b$ from left to right in this order.
5. Publicly rearrange Columns $1, 2, ..., b$ by a permutation $q = (q_1, q_2, ..., q_b)$, i.e. move Column j to Column q_j for every $j = 1, 2, ..., b$.

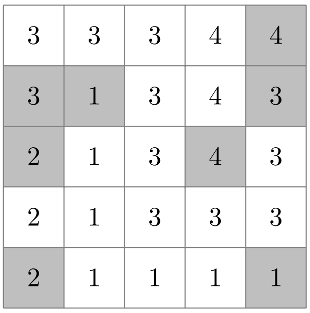
Note that since we first apply the double-scramble shuffle at Step 1, the order of Rows $2, 3, ..., a$ and the order of Columns $1, 2, ..., b$ are uniformly distributed among all possible permutations. Therefore, revealing marking cards in Steps 2 and 4 does not leak any information about the cards.

3 Our Main Protocol

3.1 Well-Designed Puzzles

For simplicity, we first consider a special case where the puzzle is well-designed.

Recall that a Numberlink grid has size $m \times n$, and has k pairs of numbers $1, 2, ..., k$ written on it. In the solution, for each path joining two terminal cells with a number x, we write a number x on every cell that the path passes through (see Figure 3). Since this is a well-designed puzzle, every path is simple and every cell has a number on it.



3	3	3	4	4
3	1	3	4	3
2	1	3	4	3
2	1	3	3	3
2	1	1	1	1

Figure 3 Transformation of the solution of the puzzle in Figure 1, with gray cells being terminal cells.

The intuition of our protocol is that the prover P will try to convince the verifier V that

1. every terminal cell has exactly one adjacent cell with the same number, and
2. every non-terminal cell has exactly two adjacent cells with the same number.

First, for each terminal cell with a number x, P publicly puts a sequence of cards $E_k(x)$ on it. Then, for each non-terminal cell with a number x, P secretly puts a sequence of cards $E_k(x)$ on it.

The verification phase for each terminal cell c works as follows.

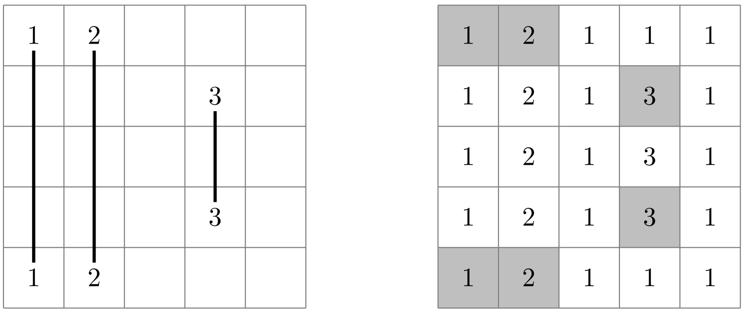
1. Publicly construct a matrix of cards in the following way: put the sequence on c into Row 1, then put the sequence on each adjacent cell to c in any order into each of the next four (or three, or two, if c is on the edge or at the corner) rows. Finally, put the number cards to complete the matrix $D(5,k)$ (or $D(4,k)$, or $D(3,k)$, for the edge or corner case).
2. Apply the double-scramble shuffle to the matrix.
3. Publicly turn over all encoding cards in Row 1. Locate the position of a $\boxed{\heartsuit}$. Suppose it is at Column j.
4. Publicly turn over all other encoding cards in Column j. If there is exactly one $\boxed{\heartsuit}$ besides the one in Row 1, then the protocol continues; otherwise V rejects and the protocol terminates.
5. Apply the rearrangement protocol to the matrix to revert the cards to their original positions, and publicly put the cards back to their corresponding cells.

The verification phase for each non-terminal cell works exactly the same as that for a terminal cell, except that in Step 4, V has to verify that there are exactly two (instead of one) $\boxed{\heartsuit}$s in Column j besides the one in Row 1.

P performs the verification phase for every cell in the grid. If every cell passes the verification, then V accepts.

In total, our protocol in the setting of a well-designed puzzle uses kmn encoding cards and $k+4$ marking cards.

▶ Remark 2. Note that in this protocol, P can convince V that the solution he/she has is valid, but cannot convince V that the puzzle is well-designed or that all cells are covered by paths in his/her solution (see Figure 4).



1	2	1	1	1
1	2	1	3	1
1	2	1	3	1
1	2	1	3	1
1	2	1	1	1

Figure 4 In a puzzle that is not well-designed, the prover P knows a solution that does not cover all cells (left), but it is possible for P to run this protocol with a non-solution (right) and get accepted.

3.2 General Puzzles

Now we consider a general case where the puzzle may not be well-designed, and the paths in our solution may not cover all cells. We can still apply a protocol similar to the one for well-designed puzzles, but with some additional tricks employed.

First, if our solution contains a non-simple path $(c_1, c_2, ..., c_t)$ with c_i being adjacent to c_j where $j > i+1$, then we replace it with a shorter path $(c_1, c_2, ...c_i, c_j, c_{j+1}..., c_t)$. We repeatedly perform this until every path in the solution becomes simple.

We put a number on each cell that is covered by a path the same way as in the well-designed setting. Then, for each cell in the i-th row and j-th column that is not covered by any path, we put a number $k+1$ on it if $i+j$ is even and put a number $k+2$ on it if $i+j$ is odd (see Figure 5). Note that by filling the numbers this way, each cell not covered by any path will have no adjacent cell with the same number.

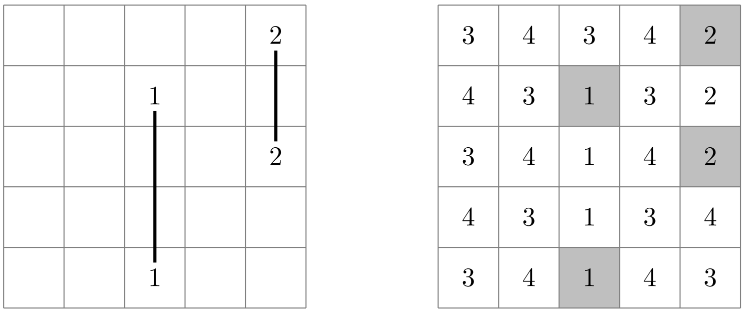


Figure 5 An example of a solution of a puzzle that is not well-designed (left), and the way we put numbers on the grid (right).

The intuition of our protocol in this setting is that the prover P will try to convince the verifier V that

1. every terminal cell has exactly one adjacent cell with the same number, and
2. every non-terminal cell either has a number $k+1$ or $k+2$, or has exactly two adjacent cells with the same number.

Since the maximum number on the grid is at most $k+2$ in this setting, we always use $E_{k+2}(x)$ instead of $E_k(x)$ to encode a number x. For each terminal cell with a number x, P publicly puts a sequence of cards $E_{k+2}(x)$ on it. Then, for each non-terminal cell with a number x, P secretly puts a sequence of cards $E_{k+2}(x)$ on it.

For each terminal cell, the verification phase works exactly the same as in the well-designed setting (except the width of the matrix will be $k+2$ instead of k). For each non-terminal cell, we add four additional rows, two encoding the number $k+1$ and two encoding the number $k+2$, to the bottom of the matrix. The formal steps for verifying each non-terminal cell c are as follows.

1. Publicly construct a matrix of cards in the following way: put the sequence on c into Row 1, then put the sequence on each adjacent cell to c into each of the next four (or three, or two, if c is on the edge or at the corner) rows in any order. Then, put the sequences $E_{k+2}(k+1)$, $E_{k+2}(k+1)$, $E_{k+2}(k+2)$, and $E_{k+2}(k+2)$ into each of the next four rows in any order. Finally, put the number cards to complete the matrix $D(9,k+2)$ (or $D(8,k+2)$, or $D(7,k+2)$, for the edge or corner case).
2. Apply the double-scramble shuffle to the matrix.
3. Publicly turn over all encoding cards in Row 1. Locate the position of a $\boxed{\heartsuit}$. Suppose it is at Column j.
4. Publicly turn over all other encoding cards in Column j. If there are exactly two $\boxed{\heartsuit}$s besides the one in Row 1, then the protocol continues; otherwise V rejects and the protocol terminates.
5. Apply the rearrangement protocol to the matrix to revert the cards to their original positions, and publicly put the cards back to their corresponding cells.

In total, our protocol in the setting of a general puzzle uses $(k+2)(mn+4)$ encoding cards and $k+10$ marking cards.

4 Proof of Security

We will prove the perfect completeness, perfect soundness, and zero-knowledge properties of our protocol. We will consider only the protocol in a general setting as it can be used for all Numberlink instances.

▶ **Lemma 3** (Perfect Completeness). *If P knows a solution of the Numberlink puzzle, then V always accepts.*

Proof. Suppose that P knows a solution that contains only simple paths, and fills numbers on the grid according to that solution.

- Consider each terminal cell c with a number $x \leq k$. There must be a path $(c_1, c_2, ..., c_t)$ starting at $c_1 = c$ and ending at c_t, the other terminal cell with the same number x. Since each cell in the grid either belongs to some path or has a number $k+1$ or $k+2$ on it, the set of all cells having a number x is exactly $\{c_1, c_2, ..., c_t\}$. We know that c_2 is adjacent to c and has a number x on it. Moreover, since the path is simple, there cannot be an index $i > 2$ such that c_i is adjacent to c. Therefore, c has exactly one adjacent cell with the same number. Since the double-scramble shuffle preserves the number of rows that encode a value equal to that of Row 1, the verification phase for c will pass.
- Consider each non-terminal cell c with a number $x \leq k$. There must be a path $(c_1, c_2, ..., c_t)$ joining two terminal cells with a number x. As previously shown, the set of all cells having a number x is exactly $\{c_1, c_2, ..., c_t\}$, so we have $c = c_i$ for some index i where $1 < i < t$. We know that c_{i-1} and c_{i+1} are adjacent to c and have a number x on them. Moreover, since the path is simple, there cannot be an index j other than $i-1$ and $i+1$ such that c_j is adjacent to c. Therefore, c has exactly two adjacent cells with the same number. Since the double-scramble shuffle preserves the number of rows that encode a value equal to that of Row 1, the verification phase for c will pass.
- Consider each non-terminal cell c with a number $x = k+1$ or $k+2$. Recall that by the way we put numbers on the cells not covered by any path, c has no adjacent cell with the same number. However, in the verification phase of c, we add four additional rows, two encoding $k+1$ and two encoding $k+2$, to the matrix. Therefore, there will be two rows that encode a value equal to that of Row 1, hence the verification phase for c will pass. ◀

▶ **Lemma 4** (Perfect Soundness). *If P does not know a solution of the Numberlink puzzle, then V always rejects.*

Proof. We will prove the contrapositive of this statement. Suppose that V accepts, meaning that the verification phase passes for every cell.

Consider each number $x \leq k$. We know that there are two terminal cells with the number x. Consider one of them, called c_1. We know from the verification phase that c_1 has exactly one adjacent cell, called c_2, with a number x. For each $i \geq 2$, if c_i is a terminal cell, then there exists a path $(c_1, c_2, ..., c_i)$ connecting the two terminal cells with the number x. Otherwise if c_i is a non-terminal cell, then we know from the verification phase that c_i has exactly two adjacent cells with the number x, one of them being c_{i-1}. We then inductively move to consider the other cell, called c_{i+1}, in the same manner. Since the path is simple, c_{i+1} must be different from any c_j with $j \leq i$. Therefore, we must eventually reach the other terminal cell, implying that there exists a path connecting the two terminal cells with the number x. Since this is true for every number $x \leq k$, there exists a path joining every pair of terminal cells with the same numbers in P's solution, which means P must know a valid solution. ◀

▶ **Lemma 5** (Zero-Knowledge). *During the verification phase, V learns nothing about P's solution of the Numberlink puzzle.*

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

Consider the verification phase of a cell c with a matrix $D(a, k+2)$ of cards ($a \in \{3, 4, 5\}$ for a terminal cell and $a \in \{7, 8, 9\}$ for a non-terminal cell). There are two steps in the verification phase where we turn over cards.

In the step where we turn over all encoding cards in Row 1, the order of Columns $1, 2, ..., k+2$ is uniformly distributed among all possible permutations due to the double-scramble shuffle, hence the $\boxed{\heartsuit}$ has an equal probability to appear at each of the $k+2$ positions. Therefore, this step can be simulated without knowing the solution.

After that, we locate the position of the $\boxed{\heartsuit}$ in Row 1 to be at Column j, and then turn over all other encoding cards in Column j. The order of Rows $2, 3, ..., a$ is uniformly distributed among all possible permutations due to the double-scramble shuffle, hence all (one or two) $\boxed{\heartsuit}$s have an equal probability to appear at each of the $a-1$ positions. Therefore, this step can be simulated without knowing the solution. ◀

5 Applications

Besides the Numberlink puzzle, our technique of transforming a graph problem into an element-counting problem can be applied to verify the existence of vertex-disjoint paths connecting k given pairs of endpoints in any undirected graph G.

In this setting, a path $(v_1, v_2, ..., v_t)$ is called *simple* if there is no i, j such that $j > i+1$ and v_i is adjacent to v_j. Similarly to the original protocol for Numberlink, if our solution contains a non-simple path $(v_1, v_2, ..., v_t)$ with v_i being adjacent to v_j where $j > i+1$, then we replace it with a shorter path $(v_1, v_2, ...v_i, v_j, v_{j+1} ..., v_t)$. We repeatedly perform this until every path in the solution becomes simple.

Suppose that the maximum degree of a vertex in G is d. We can inductively color the vertices of G with at most $d+1$ colors in linear time such that no adjacent vertices have the same color. Similarly to the protocol for Numberlink, for each path connecting the x-th given pair of endpoints, we put a number x on every vertex on that path. For each vertex v not covered by any path, we put a number $k+i$ on v if it has the i-th color in our $(d+1)$-coloring of G. By putting the numbers this way, each vertex not covered by any path will have no neighbor with the same number.

Let T be the set of vertices that appear in the list of k given pairs of endpoints. The intuition of our protocol is that the prover P will try to convince the verifier V that

1. every vertex in T has exactly one adjacent vertex with the same number, and
2. every vertex not in T either has a number greater than k, or has exactly two adjacent vertices with the same number.

Since the maximum number on the vertices is at most $k+d+1$, we use $E_{k+d+1}(x)$ to encode a number x. For each vertex $v \in T$ with a number x, P publicly puts a sequence of cards $E_{k+d+1}(x)$ on v. Then, for each vertex $v \notin T$ with a number x, P secretly puts a sequence of cards $E_{k+d+1}(x)$ on v.

The verification phase works in the same manner as in the protocol for Numberlink in a general setting. For a vertex $v \in T$, P puts the sequence on v into the first row, and the sequence on each of v's neighbors into each of the next (at most) d rows of the matrix. The verifier V has to verify that there is exactly one $\boxed{\heartsuit}$ in the same column as the $\boxed{\heartsuit}$ in Row 1. For a vertex $v \notin T$, P does the same but also puts $2(d+1)$ additional rows to the matrix, with two of them encoding each of the numbers $k+1, k+2, ..., k+d+1$. V then has to verify that there are exactly two $\boxed{\heartsuit}$s in the same column as the $\boxed{\heartsuit}$ in Row 1.

In total, this protocol uses $(k+d+1)(|V_G|+2d+2)$ encoding cards and $k+4d+3$ marking cards, where V_G is the set of vertices of G.

6 Future Work

We developed a physical protocol of zero-knowledge proof for Numberlink using $\Theta(kmn)$ cards. A challenging future work involving Numberlink is to develop a protocol of zero-knowledge proof that requires asymptotically fewer number of cards, or the one that can convince the verifier that the prover's solution contains paths that cover all cells (which is apparently a requirement in a variant of rule used in some newly developed mobile apps).

Other possible future work includes developing protocols of zero-knowledge proof for other logic puzzles, as well as exploring the method to physically verify other interesting functions.

References

1 A. Adcock, E.D. Demaine, M.L. Demaine, M.P. O'Brien, F. Reidl, F.S. Villaamil, and B.D. Sullivan. Zig-Zag Numberlink is NP-complete. *Journal of Information Processing*, 23(3):239–245, 2015. `doi:10.2197/ipsjjip.23.239`.

2 X. Bultel, J. Dreier, J.-G. Dumas, and P. Lafourcade. Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In *Proceedings of the 8th International Conference on Fun with Algorithms (FUN)*, pages 8:1–8:20, 2016. `doi:10.4230/LIPIcs.FUN.2016.8`.

3 X. Bultel, J. Dreier, J.-G. Dumas, P. Lafourcade, D. Miyahara, T. Mizuki, A. Nagao, T. Sasaki, K. Shinagawa, and H. Sone. Physical zero-knowledge proof for Makaro. In *Proceedings of the 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 111–125, 2018. `doi:10.1007/978-3-030-03232-6_8`.

4 Y.-F. Chien and W.-K. Hon. Cryptographic and physical zero-knowledge proof: From Sudoku to Nonogram. In *Proceedings of the 5th International Conference on Fun with Algorithms (FUN)*, pages 102–112, 2010. `doi:10.1007/978-3-642-13122-6_12`.

5 J.-G. Dumas, P. Lafourcade, D. Miyahara, T. Mizuki, T. Sasaki, and H. Sone. Interactive physical zero-knowledge proof for Norinori. In *Proceedings of the 25th International Computing and Combinatorics Conference (COCOON)*, pages 166–177, 2019. `doi:10.1007/978-3-030-26176-4_14`.

6 O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. *Journal of the ACM*, 38(3):691–729, 1991. `doi:10.1145/116825.116852`.

7 S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. `doi:10.1137/0218012`.

8 Google Play: Numberlink. `https://play.google.com/store/search?q=Numberlink`.

9 R. Gradwohl, M. Naor, B. Pinkas, and G.N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. In *Proceedings of the 4th International Conference on Fun with Algorithms (FUN)*, pages 166–182, 2007. `doi:10.1007/978-3-540-72914-3_16`.

10 Y. Hashimoto, K. Shinagawa, K. Nuida, M. Inamura, and G. Hanaoka. Secure grouping protocol using a deck of cards. In *Proceedings of the 10th International Conference on Information Theoretic Security (ICITS)*, pages 135–152, 2017. `doi:10.1007/978-3-319-72089-0_8`.

11 T. Ibaraki and Y. Manabe. 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)*, pages 252–257, 2016. doi:10.1109/MCSI.2016.054.

12 R. Ishikawa, E. Chida, and T. Mizuki. 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)*, pages 215–226, 2015. doi:10.1007/978-3-319-21819-9_16.

13 K. Kotsuma and Y. Takenaga. NP-completeness and enumeration of Number Link puzzle. *IEICE Technical Report*, 109(465):1–7, 2010.

14 J.F. Lynch. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter*, 5(3):31–36, 1975. doi:10.1145/1061425.1061430.

15 D. Miyahara, T. Sasaki, T. Mizuki, and H. Sone. Card-based physical zero-knowledge proof for Kakuro. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E102.A(9):1072–1078, 2019. doi:10.1587/transfun.E102.A.1072.

16 Nikoli: Numberlink. https://www.nikoli.co.jp/en/puzzles/numberlink.html.

17 T. Sasaki, T. Mizuki, and H. Sone. Card-based zero-knowledge proof for Sudoku. In *Proceedings of the 9th International Conference on Fun with Algorithms (FUN)*, pages 29:1–29:10, 2018. doi:10.4230/LIPIcs.FUN.2018.29.