How to construct physical zero-knowledge proofs for puzzles with a “single loop” condition

Pascal Lafourcade, Daiki Miyahara, Takaaki Mizuki, Léo Robert, Tatsuya Sasaki, Hideaki Sone

Theoretical Computer Science 888 (2021) 41–55

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

How to construct physical zero-knowledge proofs for puzzles with a "single loop" condition ☆

Pascal Lafourcade [a,*], Daiki Miyahara [d,c,*], Takaaki Mizuki [d,*], Léo Robert [a,*], Tatsuya Sasaki [b], Hideaki Sone [d]

[a] LIMOS, CNRS UMR 6158, University Clermont Auvergne, France
[b] Graduate School of Information Sciences, Tohoku University, Japan
[c] National Institute of Advanced Industrial Science and Technology (AIST), Japan
[d] Cyberscience Center, Tohoku University, Japan

ARTICLE INFO

Article history:
Received 29 December 2020
Received in revised form 26 June 2021
Accepted 16 July 2021
Available online 27 July 2021
Communicated by G. Persiano

Keywords:
Cryptography
Physical zero-knowledge proof
Nikoli
Masyu
Slitherlink

ABSTRACT

We propose a technique to construct physical Zero-Knowledge Proof (ZKP) protocols for puzzles that require a single loop draw feature. Our approach is based on the observation that a loop has only one hole and this property remains stable by some simple transformations. Using this trick, we can transform a simple big loop, which is visible to anyone, into the solution loop by using transformations that do not disclose any information about the solution. We illustrate our technique by applying it to construct physical ZKP protocols for two Nikoli puzzles: Slitherlink and Masyu.

1. Introduction

Introduced by Goldwasser, Micali, and Rackoff [13], *Zero-Knowledge Proof (ZKP)* systems are cryptographic protocols between a prover P and a verifier V. Given an instance $\mathcal{I}$ of a problem, only the prover P knows its solution w. The prover P wants to convince the verifier V that he/she knows w without revealing any information about w. In cryptography, the ZKPs are used to allow a party to prove that it has data without leaking any information on this data. A *zero-knowledge proof* should satisfy the following three properties:

Completeness. If P knows w, then P can convince V.

Extractability. If P does not know w, then P cannot convince V. This property involves the standard *soudness* and *proof-of-knowledge*. If there is a machine, called the *extractor*, that can interact P and extract w, then the extractability holds.

☆ This article is an extended version of the paper published at ISPEC 2019 [22]. The main difference is the presentation, a tool for ZKP for single loop, and a ZKP protocol for Masyu puzzle.

* Corresponding authors.
E-mail addresses: pascal.lafourcade@uca.fr (P. Lafourcade), daiki.miyahara.q4@alumni.tohoku.ac.jp (D. Miyahara), mizuki+tcs@tohoku.ac.jp (T. Mizuki), leo.robert@uca.fr (L. Robert).

https://doi.org/10.1016/j.tcs.2021.07.019

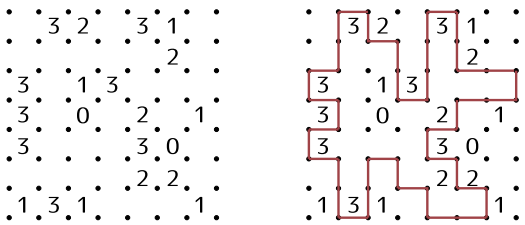


Fig. 1. Example of a standard Slitherlink challenge, and its solution.

Zero-Knowledge. V cannot obtain any information about w. Assuming a probabilistic polynomial time simulator $M(\mathcal{I})$ that can emulate the interaction between P and V but does not contain w, if it outputs of the protocol and $M(\mathcal{I})$ follow the same probability distribution, then the zero-knowledge holds.

It is well known that for any NP-complete problem, there exists an interactive ZKP protocol [12]. Later, one of the first physical ZKP protocols was introduced by Gradwohl et al. [14] for a popular puzzle, Sudoku. A player of Sudoku who knows a solution wants to prove to someone else that he/she knows the solution using only physical objects. For this, the authors utilized physical cards. Recently, more efficient ZKP protocols [35] have been proposed. They used envelopes and physical tricks to improve the original protocol. Notice that for each specific puzzle, there exists a specific ZKP; indeed, one cannot simply use a ZKP for Sudoku to be applied to Masyu.

Nikoli[1] is a Japanese company famous for designing puzzles. The list of puzzles created by Nikoli contains more than 40 different kinds of puzzles including Sudoku. We focus on two puzzles that aim at drawing a single loop: *Slitherlink* and *Masyu*.

- Slitherlink is one of the most famous pencil puzzles published in the puzzle magazine *Nikoli*. It was introduced in 1989 in the 26th of Nikoli's Puzzle Times. It is also known as *Loop-the-Loop*. It is explained on Nikoli's website as follows: "*Getting the loop right is absorbing and addictive. Watch out not to get lost in Slitherlink. It's amazing to see how endless patterns can be made using only four numbers (0, 1, 2 and 3)*". The puzzle instance consists of lattice-like dots where some squares contain numbers between 0 and 3. The goal of the puzzle is to draw lines that satisfy the following rules [1]:

  1. **Loop Rule:** Connect vertically/horizontally adjacent dots with lines to make a single loop and the loop never crosses itself and never branches off.
  2. **Numbers Rule:** Each number indicates the number of lines that should surround its square, while empty squares may be surrounded with any number of lines.

  Fig. 1 shows an example of a Slitherlink puzzle and its solution; one can easily verify that all conditions are satisfied. Slitherlink was proven to be NP-complete in [40] and other variants in [21].
- Masyu was invented by Ryuou Yano, originally called *Pearl Necklace*, and it was introduced in Puzzle Communication Nikoli #84. The goal of this puzzle is to draw a single continuous non-intersecting loop according to some constraints that depend on "black" (filled) or "white" (empty) circles placed on a rectangular grid of squares. The loop can only have 90-degree turns and has to pass through all circles as follows: White circles must be traveled straight through, but the loop must turn in the previous and/or next cell in its path. Black circles must be turned upon, but the loop must travel straight through the next and previous cells in its path. More precisely, it consists of rectangular lattice-like dots where some dots are replaced by black or white circles. We present an example of an initial grid in Fig. 2 and its solution in Fig. 3. The goal is to draw lines that satisfy the following rules:

  1. **Loop Rule:** Connect vertically/horizontally adjacent dots with lines to make a single loop that never crosses itself.
  2. **Through Circle Rule:** The loop must pass through all circles.
  3. **Black Circle Rule:** The entering line must be perpendicular to the exiting line. In addition, the loop must go straight for the two next dots. In Fig. 4, we show a valid path passing a black circle and in Fig. 5 an invalid one.
  4. **White Circle Rule:** The entering and exiting line must form a straight line. In addition, the line of at least one extremity must be perpendicular. In Fig. 6, we show examples of valid paths passing white circles and in Fig. 7 one invalid.

  In 2002, Erich Freidman proved that this puzzle is NP-complete in [11,17].

Since these two puzzles are NP-complete, we can apply the technique of O. Goldreich [12] to construct a ZKP. Hence our aim is to propose a physical ZKP protocol for puzzles that require to draw a single continuous non-intersecting loop.

[1] http://www.nikoli.com/.

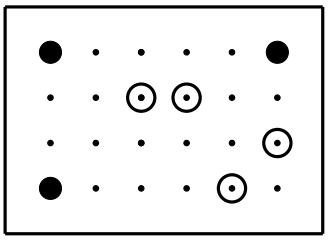


Fig. 2. Initial grid of Masyu puzzle.

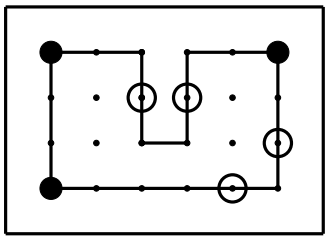


Fig. 3. Solution of Masyu puzzle.

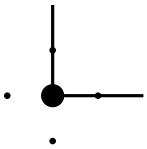


Fig. 4. Valid black circle.

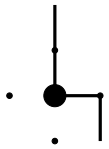


Fig. 5. Invalid black circle.

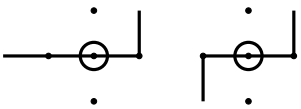


Fig. 6. Valid white circle.

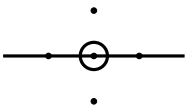


Fig. 7. Invalid white circle.

Taking into account this feature is clearly a challenge that was not present in the previous physical ZKP protocols for Nikoli's puzzles [4–6,10,14].

Contributions We introduce a technique to construct a ZKP protocol for a puzzle where constructing a single loop is one of the requirements of the solution. The difficulty is to avoid leaking any information regarding the solution to the verifier. For this, we use a topological point of view; more precisely, we use the notion of homology that defines and categorizes holes in a manifold. The main idea is that after any continuous transformations, the number of holes always remains the same. Using this simple idea, we construct transformations that preserve the number of loops in the solution. First, the verifier checks that the initial configuration has only a single big loop. Then, by transforming in several steps this trivial big loop into the solution, the prover convinces step after step that the solution has only one loop at the end by proving that the transformation does not break the loop or introduce an extra hole. This construction is applied to Slitherlink and Masyu, two Nikoli puzzles that aim at drawing a single loop. This technique can be used for any other puzzles that require such type of features in their rules.

Related works Since Gradwohl et al. [14] introduced the first physical ZKP protocol for Sudoku, physical ZKPs for other puzzles have been proposed, e.g., Nonogram [6], Akari, Takuzu, Kakuro, Kenken [5], Makaro [4], Norinori [10], Numberlink [33], Juosan [24], Suguru [31], and Ripple Effect [34]. Almost all these ZKPs deal with numbers.

There are many physical cryptographic protocols without relying on computers: such examples are a PEZ dispenser [2,3], tamper-evident seals [28], visual secret sharing schemes [8], and a deck of cards [9]. Among them, *card-based cryptography* using a deck of cards has been widely studied. Especially, for secure computation of the logical AND function, the number of required cards have been reduced [7,18,20,25–27,29,37], and necessary and sufficient numbers of cards and/or shuffles have been provided [16,18,20,36]. Some recent studies applied card-based techniques to propose practical implementations of Yao's millionaire protocol [23] and secure ranking computation [38].

However, these works do not deal with proving the topological feature of having a single loop in the solution.

Outline In Section 2, we introduce some notations and useful sub-protocols used for physical ZKP based on cards. In Section 3, we explain how to make a ZKP for a single loop. In Section 4, we describe our ZKP protocol for Slitherlink. In Section 5, we show the security proofs of our ZKP for Slitherlink. In Section 6, we present our ZKP protocol for Masyu. In the next section, we give the security proofs for Masyu before concluding.

2. Preliminaries

In this section, we first introduce some notation of a deck of cards and shuffling actions used in our constructions. Then, we introduce useful sub-protocols.

2.1. Notations

We use a sequence of physical cards denoted by $\boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\heartsuit}\boxed{\heartsuit}\cdots\boxed{1}\boxed{2}\cdots$; the black $\boxed{\clubsuit}$ and red $\boxed{\heartsuit}$ cards are called *binary cards* and $\boxed{1}\boxed{2}\cdots$ are *numbered cards*. The backs of all cards are identical and denoted by $\boxed{?}$. As seen later, we use binary cards to encode the existence of a line while numbered cards are used for rearranging the positions of cards.

Encoding Boolean values are encoded with two binary cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. Two face-down cards $\boxed{?}\boxed{?}$ encoding 0 and 1 are called a *0-commitment* and a *1-commitment*, which are denoted by $\boxed{0}$ and $\boxed{1}$, respectively.

In our protocols, a 0-commitment placed on a gap between two adjacent dots means that there is no line on the gap, and a 1-commitment means that there is a line on the gap. Following this encoding, one can represent a loop by putting a commitment on each gap. Note that given an x-commitment for $x \in \{0, 1\}$, the negated $\overline{x}$-commitment can be easily obtained by swapping the two cards of the commitment.

Shuffle Let S_m denote the symmetric group of degree m. Given a sequence of m face-down cards $(c_1, c_2, \ldots, c_m)$, a *shuffle* outputs a sequence of face-down cards $\left(c_{r^{-1}(1)}, c_{r^{-1}(2)}, \ldots, c_{r^{-1}(m)}\right)$, where $r \in S_m$ is a uniformly distributed random permutation.

Pile-shifting shuffle This is to *cyclically* shuffle piles of cards [30]; given m piles, each of which consists of the same number of face-down cards, denoted by $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m)$, applying a *pile-shifting shuffle* (denoted by $< \cdot | \cdots | \cdot >$) outputs $(\boldsymbol{p}_{s+1}, \boldsymbol{p}_{s+2}, \ldots, \boldsymbol{p}_{s+m})$:

$$\left\langle \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_1} \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_2} \cdots \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_m} \right\rangle \rightarrow \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_{s+1}} \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_{s+2}} \cdots \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_{s+m}},$$

where s is uniformly and randomly chosen from $\mathbb{Z}/m\mathbb{Z}$. A possible implementation of a pile-shifting shuffle is the use of physical cases that can store a pile of cards, such as boxes and envelopes; a player (or players) cyclically shuffle them by hand until everyone lost track of the offset.

If each pile consists of one face-down card, then the shuffle action is called a *random cut*. An implementation of random cut has been studied using the so-called Hindu Cut without any additional tool [39].

Pile-scramble shuffle This is a shuffle operation on a sequence of m piles $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m)$ [15]. For such a sequence of piles, applying a *pile-scramble shuffle* outputs $(\boldsymbol{p}_{r^{-1}(1)}, \boldsymbol{p}_{r^{-1}(2)}, \ldots, \boldsymbol{p}_{r^{-1}(m)})$, where $r \in S_m$ is a uniformly distributed random permutation. A pile-scramble shuffle uses similar materials as a pile-shifting shuffle but its operation is similar to a shuffle.

Chosen pile cut The *chosen pile cut* [19] enables a prover to choose a pile $\boldsymbol{p}_i$ from m piles $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m)$ without revealing i to a verifier. The chosen pile cut proceeds as follows, given m piles along with m additional cards:

1. The prover P holds $m-1$ $\boxed{\clubsuit}$s and one $\boxed{\heartsuit}$. Then, P places m face-down cards below the piles such that only the i-th card is $\boxed{\heartsuit}$:

$$\begin{array}{cccccccccc} \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_1} & \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_2} & \cdots & \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_{i-1}} & \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_i} & \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_{i+1}} & \cdots & \underbrace{\boxed{?}\boxed{?}}_{\boldsymbol{p}_m} \\ \boxed{?}_{\clubsuit} & \boxed{?}_{\clubsuit} & \cdots & \boxed{?}_{\clubsuit} & \boxed{?}_{\heartsuit} & \boxed{?}_{\clubsuit} & \cdots & \boxed{?}_{\clubsuit} \end{array}$$

2. Regarding the cards in the same column as a pile, apply a pile-shifting shuffle to the piles:

$$\left\langle \begin{array}{c|c|c|c} \underbrace{\boxed{?}}_{\boldsymbol{p}_1} & \underbrace{\boxed{?}}_{\boldsymbol{p}_2} & \cdots & \underbrace{\boxed{?}}_{\boldsymbol{p}_m} \\ \boxed{?} & \boxed{?} & & \boxed{?} \end{array} \right\rangle \rightarrow \begin{array}{cccc} \underbrace{\boxed{?}}_{\boldsymbol{p}_{s+1}} & \underbrace{\boxed{?}}_{\boldsymbol{p}_{s+2}} & \cdots & \underbrace{\boxed{?}}_{\boldsymbol{p}_{s+m}} \\ \boxed{?} & \boxed{?} & \cdots & \boxed{?} \end{array},$$

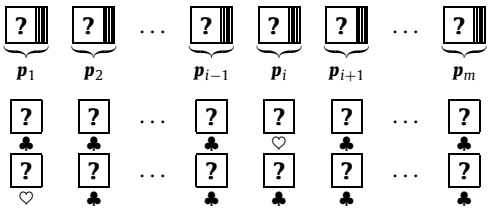
where s is generated uniformly at random from $\mathbb{Z}/m\mathbb{Z}$ by this shuffle action.

3. Reveal all the cards in the second row; then, one $\boxed{\heartsuit}$ should appear in the $(i+s)$-th, and the pile above the revealed $\boxed{\heartsuit}$ is $\boldsymbol{p}_i$, and hence, P can choose the desired $\boldsymbol{p}_i$. Note that no information about i (i.e., the index of the chosen pile) is leaked because a red $\boxed{\heartsuit}$ appears in the $(i+s)$-th in the second row.

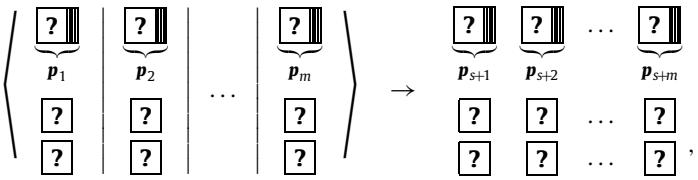
2.2. Sub-protocols

Chosen pile protocol This is an extended version of the chosen pile cut [19] explained above. Given m piles $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m)$ with $2m$ additional cards, this protocol enables P to choose the i-th pile $\boldsymbol{p}_i$ and regenerate the original sequence of m piles.

1. In the same way as Step 1 of the chosen pile cut, the prover P places m face-down cards such that only the i-th is $\boxed{\heartsuit}$. We further put m face-down cards below the cards such that only the first card is $\boxed{\heartsuit}$:

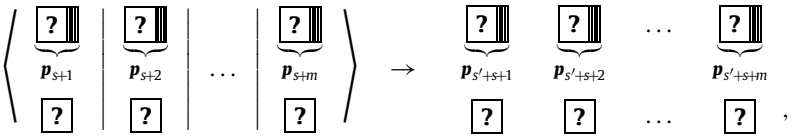


2. Similar to Step 2 of the chosen pile cut, apply a pile-shifting shuffle to the sequence of piles:



where s is generated uniformly at random from $\mathbb{Z}/m\mathbb{Z}$.
3. Similar to Step 3 of the chosen pile cut, reveal all the cards in the second row; then, one $\boxed{\heartsuit}$ should appear in the $(i+s)$-th, and the pile above the revealed $\boxed{\heartsuit}$ is the i-th pile (and hence, P can choose $\boldsymbol{p}_i$).

   When this protocol is invoked, the chosen pile will be picked, and certain operations will be applied to the chosen pile. Then, the chosen pile is placed back to the $(i+s)$-th in the sequence.
4. Remove the revealed cards, i.e., the cards in the second row. Then, apply a pile-shifting shuffle:

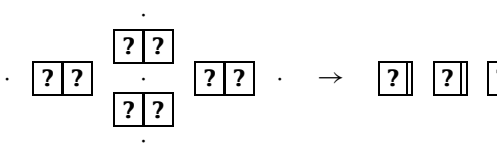


where s' is generated uniformly at random from $\mathbb{Z}/m\mathbb{Z}$.
5. Reveal all the cards in the second row; then, one $\boxed{\heartsuit}$ should appear in the $(1+s+s')$-th, and the pile above the revealed $\boxed{\heartsuit}$ is $\boldsymbol{p}_1$. Therefore, by shifting the sequence of piles (such that $\boldsymbol{p}_1$ becomes the first pile in the sequence), we can obtain a sequence of piles whose order is the same as the original one without revealing any information about the order of the input sequence.

Verifying-degree protocol This protocol enables P to convince V that the "degree" of a target vertex (dot) is not four without revealing anything beyond that. Here, the *degree* means the number of 1-commitments placed around a target vertex. The idea behind this protocol is that it suffices for P to choose one 0-commitment around the target vertex by using the chosen pile protocol explained above (when only P knows what the four commitments around the target are).

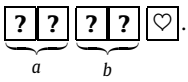
The verifying-degree protocol proceeds as follows.

1. Given four commitments placed around the target vertex, regard them as a sequence of four commitments:

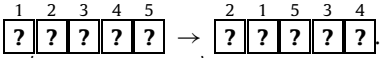
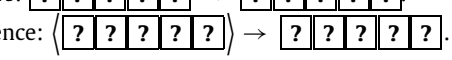
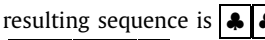


2. By using the chosen pile protocol, P chooses one of the 0-commitments. Then, reveal the chosen pile, namely the chosen 0-commitment, to prove that it is surely 0. Now, V is convinced that the degree of the target vertex is not four. Then, turn over the revealed cards. Note that no information about the four commitments is leaked because a 0-commitment is always revealed in this step.
3. V performs the remaining steps in the chosen pile protocol. Then, place all the commitments back to their original positions.

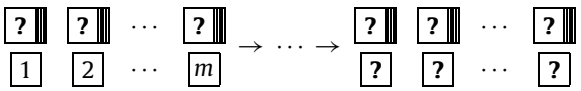
Five-card trick This sub-protocol will be used for the Masyu puzzle. Given two commitments to $a, b \in \{0, 1\}$ (along with a red card $\boxed{\heartsuit}$), the five-card trick [9] starts by adding an extra card as follows:



The sub-protocol proceeds as follows to compute $a \wedge b$:

1. Rearrange the sequence as follows: 
2. Apply a random cut to the sequence: 
3. Reveal the sequence.
   (a) If the resulting sequence is  (up to cyclic shifts), the output is $a \wedge b = 1$.
   (b) If it is  (up to cyclic shifts), the output is $a \wedge b = 0$.

Input-preserving function evaluation technique Assume that we have a protocol (e.g., the five-card trick) to evaluate some function with m input piles of cards. The *input-preserving function evaluation technique* enables us to obtain the m input piles again after evaluating some function by using m number cards, as follows [24].

1. Attach a corresponding numbered card to each of m input piles:
   
   Then, together with the added numbered cards, execute a designated protocol to evaluate some function.
2. Apply a pile-scramble shuffle to the sequence of piles.
3. Reveal only the numbered cards. Then, rearrange the sequence of piles so that the revealed numbered cards become in ascending order to obtain m input piles.

3. How to make a single loop

As mentioned before, both Slitherlink and Masyu have the same constraint: making a single loop. In this section, we present a generic way for the prover P to convince the verifier V that the constraint is satisfied. Let us first show that achieving this is somewhat difficult.

3.1. Naive approach does not work well

By using binary cards and the encoding rule shown in Section 2, we can consider a naive approach to construct physical ZKP protocols for both Slitherlink and Masyu, as follows.

Naive setup. P places a commitment on each gap according to the solution.
Verification. V verifies that the placement of the commitments satisfies all the constraints.

To the best of our knowledge, all the existing ZKP protocols for puzzles were constructed based on this approach. As one might imagine, verifying the Numbers Rule in Slitherlink is relatively simple, which will be seen later. However, how can V verify the Loop Rule? One might consider the use of a Boolean circuit, as follows.

1. For each gap, numbered i, consider a Boolean variable $c_i \in \{0, 1\}$. That is, c_i means the existence of a line on the gap.
2. Construct a Boolean circuit C that outputs 1 if and only if all the variables $c_1, c_2, \ldots$, represent a single loop.
3. Perform a secure computation of C using all the commitments placed on the gaps; if it outputs 0, V rejects it.

With ignoring the efficiency, the above approach would work. However, we note that constructing C is not feasible; C could be large and complicated. It means that a different approach needs.

3.2. Our approach: topology-preserving computation

As we mentioned in Section 1, our idea is to transform a single loop into another one while preserving the properties of the single loop without leaking any information about its specific shape. We call this transformation the *topology-preserving computation*. This will be used in both our ZKP protocols for Slitherlink and Masyu.

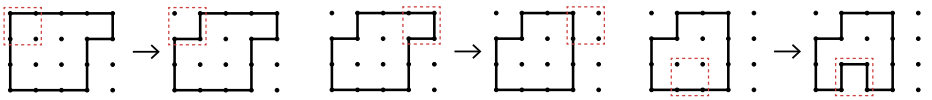
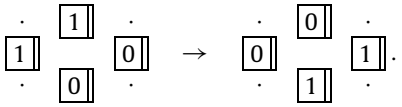
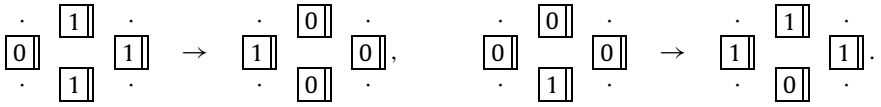


Fig. 8. The three transformations.

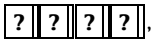
Let us explain the idea behind our topology-preserving computation. This protocol changes a given loop into another loop by applying one of the three transformations given in Fig. 8. Each transformation changes the lines surrounding a square, represented by dash line in Fig. 8. Remember that a line is expressed by a commitment (i.e., two face-down binary cards) in our protocols. Therefore, for example, the (2,2)-transformation means



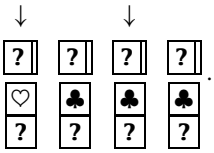
This can be performed by swapping the two cards of each commitment. (Remember that swapping the two cards performs negation of a commitment.) The (3,1)-transformation and the (1,3)-transformation can also be performed by swapping the two cards of each commitment:



Now, suppose that P wants to apply one of the three transformations while the applied transformation is hidden from V. Furthermore, P needs to show that the commitments around a target square are "transformable." Note that the three transformations are applicable to the four commitments around a square if and only if a 0-commitment facing a 1-commitment exists among the four commitments.

Given four commitments around a target square  , the topology-preserving computation proceeds as follows.

1. P chooses a 0-commitment facing a 1-commitment using the chosen pile protocol.
2. V reveals the chosen commitment as well as the commitment that is two piles away from it:



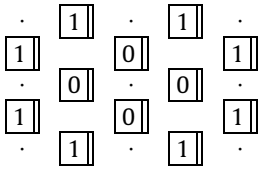
   Then, V checks that the two commitments are a 0-commitment and a 1-commitment to be convinced that any transformation can be applied.
3. After turning over all the opened cards of the commitments, V performs the remaining steps of the chosen pile protocol to place all the commitments back to their original positions.
4. Swap the two cards of each of the four commitments. (Remember that this results in negating all the four commitments, and hence, one of the three transformations has been applied.)
5. V applies the verifying-degree protocol to each of the four dots of the target square. Then, V is convinced that no dots of degree four has arisen as the result of transformation. As will be proved in Section 5, this guarantees that the loop was not split and thus, it remains a single loop.

As Lemma 2 in Section 5, we will prove that the resulting placement of 1-commitments after the topology-preserving computation always represents a single loop.

3.3. Input phase

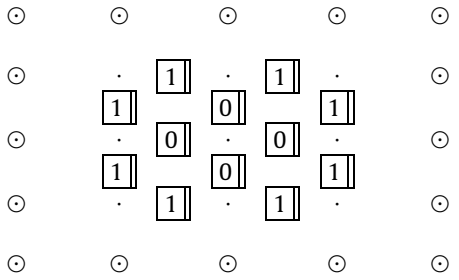
We describe a generic procedure for an input before applying the topology-preserving computation, which aims to convince a verifier V that the commitments placed by a prover P form a single, non-intersecting loop. Note that this phase is applied for Slitherlink and Masyu, but it can be applied to any puzzle where the constraint of a single non-intersecting loop is mandatory.

The verifier V puts a 1-commitment on every gap on the boundary of the puzzle board and 0-commitments on all the other gaps. This placement corresponds to the single loop with the same size as the board. The following is an example of the placement of a (2×2)-square puzzle board:

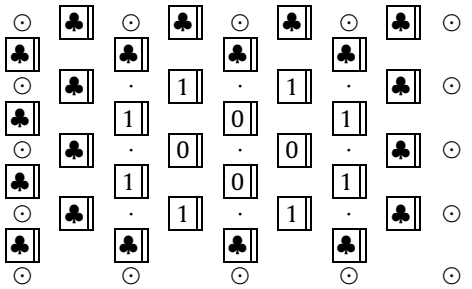


P applies the topology-preserving computation to these commitments to transform the shape of the loop into the solution. To hide the position of the target square, we make a sequence of piles from the placed cards, pick the four target commitments using the chosen pile protocol, and apply the topology-preserving computation. To properly pick the four commitments, a sequence of piles is formed, as follows.

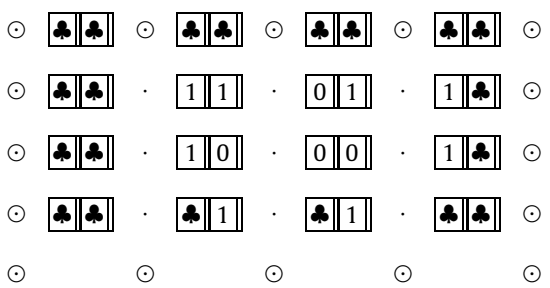
First, we expand the puzzle board by adding dots around the original board. The expanded dots are denoted by $\odot$.



Note that the expanded area is unrelated to the actual puzzle board. V puts "dummy" commitments on the gaps at the expanded area other than the right and the bottom ends. Each dummy commitment consists of two black cards $\boxed{\clubsuit}\boxed{\clubsuit}$ to prevent the loop from spreading over the expanded area. We denote a dummy commitment by $\boxed{\clubsuit}$.



Next, V makes a sequence of 4-card piles as follows. For each square, V first makes a pile from the commitments placed on the left and the top.[2]



(1)

Then, pick 4-card piles from top to bottom:

$$\boxed{\clubsuit}\boxed{\clubsuit}\quad \boxed{\clubsuit}\boxed{\clubsuit}\quad \boxed{\clubsuit}\boxed{\clubsuit}\quad \ldots\quad \boxed{\clubsuit}\boxed{1}\quad \boxed{\clubsuit}\boxed{1}\quad \boxed{\clubsuit}\boxed{\clubsuit},$$

to make a sequence of piles: $\boxed{?}\quad \boxed{?}\quad \boxed{?}\quad \ldots\quad \boxed{?}\quad \boxed{?}\quad \boxed{?}$.

[2] The commitment on the gap between each vertically consecutive dots is placed on the commitment on its upper right.

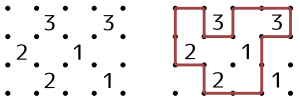


Fig. 9. Small example of Slitherlink challenge, and its solution.

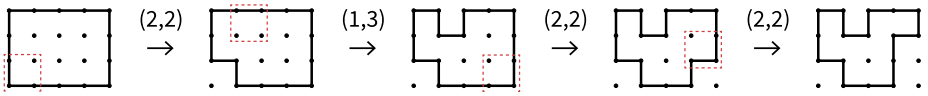


Fig. 10. Transformation process.

4. Zero-knowledge proof for Slitherlink

In this section, we show our physical zero-knowledge proof protocol for Slitherlink. The outline of our protocol is as follows:

Input Phase: As described in Section 3.3, the verifier V puts commitments on every gap.
Topology-Preserving Computation Phase: The prover P transforms the big loop into the solution loop. After this phase, V is convinced that the placement of 1-commitments satisfies the Loop Rule of Slitherlink without revealing any information about its shape.
Verification Phase: V verifies that the placement of 1-commitments satisfies the Numbers Rule of Slitherlink.

4.1. Our construction

The main idea behind our protocol is that the prover P can transform a single loop into a solution loop while keeping the properties of the loop (to convince the verifier V that the Loop Rule is satisfied). Let us consider a puzzle instance shown in Fig. 9 as an example. Our protocol transforms the loop as illustrated in Fig. 10. The full description of our proposed protocol is as follows.

4.1.1. Input phase

As described in Section 3.3, the verifier V puts a 1-commitment on every gap on the boundary of the puzzle board and 0-commitments on all the other gaps. This placement corresponds to the single loop with the same size as the board. Next, make a sequence of 4-card piles, as described in Section 3.3.

4.1.2. Topology-preserving computation phase

In this phase, P applies the transformations (explained in Section 3.2) to step-wise change the loop. Let n be the size of the puzzle instance, namely the number of squares on the puzzle board. Then, note that P can make the solution loop by at most n transformations.

1. P applies the following exactly $n-1$ times such that either the resulting loop is already the solution, or one more transformation will end up the solution.[3] (This is possible because successive two transformations (of the same) to the same square keep the loop unchanged.)

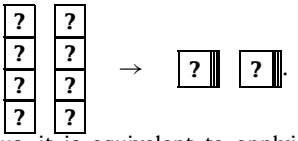
   (a) P applies the chosen pile protocol to the sequence of 4-card piles: P picks a 4-card pile composed of the left and top edges of the square that P wants to transform. The remaining edges can be picked by counting the distance from the chosen pile.[4]
   (b) P applies the topology-preserving computation to the four picked commitments.
   (c) V performs the remaining steps of the chosen pile protocol to place the cards back to their original positions.

2. P chooses either to apply one more transformation or not to change the solution loop so that V does not learn which action occurs:

   (a) Similarly to Step 1 (a) above, P picks four commitments around the target square.
   (b) By applying Steps 1 to 3 of the topology-preserving computation, V confirms that any transformation is applicable.

[3] More precisely, if the size of the solution is odd, then the resulting loop will be the solution after Step 1; otherwise, one more transformation will end up the solution.

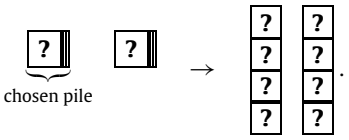
[4] In the example of the board (1) in Section 3.3, the bottom edge corresponds to the pile that is four piles away from the chosen pile. Note that the distance between any two piles is fixed because only a pile-shifting shuffle is applied.

(c) V arranges the four commitments vertically and makes a pile from each column:  .

Note that swapping the two piles results in negating each commitment. Thus, it is equivalent to applying a transformation.

(d) Using the chosen pile cut, if P wants to transform the target square, then P chooses the right pile; otherwise, the left pile is chosen.

(e) Rearrange the cards vertically such that the chosen pile is placed on the left:

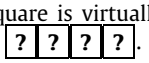


(f) V makes four commitments from each row, performs the remaining steps of the chosen pile protocol, and places each commitment back to their original positions.

3. Finally, all cards are placed on the puzzle board and the cards in the dummy area are removed.

4.1.3. Numbers verification phase

V is now convinced that the placement of 1-commitments represents a single loop and it never branches off (the Loop Rule). Finally, V verifies that the placement satisfies the Numbers Rule of Slitherlink. That is, the number on each square is equal to the number of lines surrounding it. The verification proceeds as follows, where we virtually assume that the board is colored like a checkered pattern so that all squares in the first row are alternation of blue and yellow, those in the second row are alternation of yellow and blue, and so on.

1. V picks all the left cards (if the square is virtually blue) or all the right cards (if the square is yellow) of four commitments around a numbered square:  . Let m denote that number.
2. P shuffles the four cards.
3. V reveals the four cards.

- If V picked all the left cards of four commitments in Step 1, V checks that the number of red cards  is equal to m.
- If V picked all the right cards, V checks that the number of black cards  is equal to m.

4. Apply Steps 1 to 3 to all the other numbered squares. (Note that every commitment is related to exactly one blue square and one yellow square.)

Performance evaluation When we have a $p \times q$ board, our protocol uses $4(p+2)(q+2)$ cards in the Input phase and $2(p+2)(q+2)$ cards in the Topology-Preserving Computation phase: $6(p+2)(q+2)$ cards in total. Note that in the chosen pile protocol, cards placed in the second and third rows can be reused after revealing them.

5. Security proofs for our construction

We show that our construction for Slitherlink satisfies the completeness, extractability, and zero-knowledge properties.

Completeness Note that P uses only (3,1)-, (1,3)-, and (2,2)-transformations in the topology-preserving computation to transform a single loop into the shape of the solution. We now prove that this is possible in Theorem 1.

Theorem 1. *Let n be the number of squares in the puzzle instance (namely, the big loop), and let k be the number of squares inside its solution loop. By applying a transformation to the loop exactly $n-k$ times, the big loop can be transformed into the solution loop.*

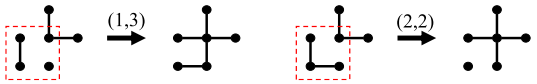
To prove Theorem 1, we first show Lemmas 1 and 2.

Lemma 1. *After the topology-preserving computation, the resulting placement of 1-commitments represents a single loop.*

Proof. Remember Steps 1 and 5 in the topology-preserving computation in Section 3.2: Due to Step 1, the target square is guaranteed to be none of the following two ones (up to rotations).

That is, one of the (2,2)-, (3,1)-, and (1,3)-transformations is always applied to the target square.

Due to the execution of the verifying-degree protocol in Step 5, the following two transformations that make a loop split cannot occur (up to rotations).

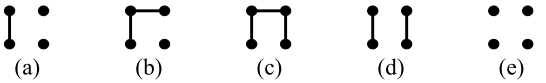


Therefore, it remains a single loop. We note that a dot of degree three (or one) cannot exist in a loop. Since the (2,2)-, (3,1)-, and (1,3)-transformations affect the lines between each of four dots (in a square), the resulting degrees of the four dots cannot be odd (after Step 4). Therefore, it suffices to verify that a dot of degree four does not exist in the target square in Step 5. □

Lemma 2. *For any single loop, at least one of the (3,1)-, (1,3)-, and (2,2)-transformations exists, which increases the number of squares inside the loop by exactly one without causing a degree four.*

Proof. Consider a single loop; let ℓ be the number of squares inside the loop. To prove this lemma, we show that there always exists a square on the board such that the (3,1)-, (1,3)-, or (2,2)-transformation can be applied to the square such that ℓ increases without causing a degree four.

If $\ell \leq 2$, then the (1,3)-transformation increases the number of squares by one. Thus, one may assume that $\ell \geq 3$. Then, any square outside the loop can be classified in one of the following five types (up to rotations):



If none of types (a), (b), and (c) exists, we cannot have a loop because only squares of types (d) and (e) cannot make a loop. Therefore, at least one square of type (a), (b), or (c) must exist outside the loop.

When the loop is convex, applying the (3,1)-, (1,3)-, or (2,2)-transformation to such an external square results in increasing ℓ by one without causing a vertex of degree four. When the loop is not convex, there exists a square of type (b) (where the dot having no line is of degree zero), or (c) inside the convex hull, and similarly, applying the transformation results in increasing ℓ by one without causing a degree four. □

By these lemmas, Theorem 1 can be proved.

Proof of Theorem 1. By Lemmas 1 and 2, we can always increase the number of squares inside the solution loop by a transformation. Therefore, we can repeat the transformation so that the solution loop becomes the big loop. This means that, conversely, the big loop can be transformed into the solution loop by applying the (3,1)-, (1,3)-, or (2,2)-transformation exactly $n - k$ times. □

Extractability Intuitively, the extractability holds because after executing the Topology-preserving Computation phase, the placement of 1-commitments encodes exactly a solution P knows. If the placement passes the verification phase, then it means that the placement satisfies the Numbers Rules (and the Loop Rule), and hence, it is the solution (and P knows the solution).

More formally, to prove the extractability, we show that any shape that does not satisfy the Loop Rule or Numbers Rule, is always rejected during the protocol, and there is an extractor that can extract a solution.

Theorem 2. *If the prover does not know a solution for a given Slitherlink puzzle, then the verifier rejects regardless of the prover's behavior.*

To prove Theorem 2, let us first show that the resulting loop after the Topology-preserving Computation phase always satisfies the Loop Rule (as in Lemma 1) and any single loop that does not satisfy the Numbers Rule is always rejected in the verification phase (as in Lemma 3). Therefore, any single loop except for a solution is always rejected.

Lemma 3. *Any (single) loop that does not satisfy the Numbers Rule is always rejected in the verification phase.*

Proof. Consider any (single) loop that does not satisfy the Numbers Rule, i.e., there are four commitments surrounding a numbered square such that the number of 1-commitments among them is not equal to the number. Due to Step 3 in the verification phase, all the left (or right) cards of four commitments are turned over (after shuffling them), and hence, the number of 1-commitments is revealed. This means that the verifier can always reject any (single) loop that does not satisfy the Numbers Rule. □

Proof of Theorem 2. By Lemma 1, the resulting loop after the Topology-preserving Computation phase is always single, i.e., it satisfies the Loop Rule. By Lemma 3, if it does not satisfy the Numbers Rule, the verifier always rejects it in the verification phase. That is, any loop except for the solution cannot go through the verification phase.

The extractor for this protocol can be simply constructed: after executing the Topology-preserving Computation phase, the extractor takes all the face-down cards on the board and look at them to obtain a solution. □

Zero-knowledge In our construction, all cards to be opened have been shuffled before being opened. Therefore, all distributions of opened cards can be simulated by a simulator $M(\mathcal{I})$ who does not know the solution. For example, at Step 2 in the verification phase, a shuffle has been applied to the opened commitments; thus, this is indistinguishable from a simulation putting randomly 1-commitments such that the number of them is equal to the number of the squares.

6. Zero-knowledge proof for Masyu

In this section, we construct a physical zero-knowledge proof protocol for Masyu. The outline of our protocol is as follows:

Input Phase: The verifier V puts a 1-commitment (i.e., two face-down cards encoding 1) on every gap on the boundary of the puzzle board and 0-commitments on all the remaining gaps. In other words, V creates a single big loop whose size is the same as the board.

Topology-Preserving Computation Phase: The prover P transforms the shape of the loop according to the solution. After this phase, V is convinced that the placement of 1-commitments satisfies the Loop Rule of Masyu without the disclosure of any information about the shape.

Crossing Verification Phase: The verifier V verifies that the placement of 1-commitments satisfies all the remaining rules.

6.1. Our construction

The main idea behind this protocol is that V creates a big loop (on the edge of the grid) and then P transforms the loop according to the solution. Once V is convinced of the Loop Rule, the verification phase is run, leading to convincing V that the black and white circles satisfy their respective constraints.

Input phase As described in Section 3.3, V prepares a placement corresponding to the single loop with the same size as the board.

Topology-preserving computation phase In this part of the protocol, P transforms a big loop into the solution loop. The process is exactly the same as in Section 4.1.2. In addition, V replaces each dummy commitment $\boxed{\clubsuit\|}$ with a 0-commitment $\boxed{0\|}$ for the verification.

Crossing verification phase The verifier V is now convinced that the placement of 1-commitments forms a non-intersecting single loop (the Loop Rule). Now, V needs to verify that the placement satisfies the rules about circles. This verification is done during a single phase.

We divide the phase in two cases: one for white circles and one for black circles.

Black circle: We want to show that V can verify that there is a line passing through a black circle, and the entering line is perpendicular to the exiting line. In addition, V can verify that the next lines of entering and exiting lines are straight.

At this step we only pick the left card of each commitment. Thus, the presence of a line is encoded as $\boxed{\heartsuit}$ and the absence is $\boxed{\clubsuit}$. We want to show that the line passing through a black circle forms a perpendicular shape and that the line continues in straight line for the two extremities.

Consider the center of the black circle of coordinate $(0, 0)$. The left dot is then $(-1, 0)$, the bottom dot is $(0, -1)$, and so on. The following verification is done for each black circle of the grid.

1. V picks the left cards of the commitments between the $(-1, 0)$ dot horizontally (namely, the two commitments to the left of the black circle) to form a pile. V keeps the closest card from the circle at the beginning of the pile. Do the same for each direction i.e., commitments between $(1, 0)$, $(0, 1)$, $(0, -1)$.[5]

   At this point, there are four piles (one for each direction) composed of two cards where the first card of each pile is the closest from the black circle.
2. They do the following steps enhanced by the input-preserving function evaluation technique shown in Section 2.2. That is, the above four piles will go back to their original positions after the last step.
3. V applies a pile-shifting shuffle to those four piles.
4. P puts the first card of each pile in a row, and does the same for the second cards.

- V reveals the first row (namely, the closest commitments from the circle); a valid configuration is (up to a cyclic shift): $\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$. This verifies that the line forms a perpendicular shape.
- Then V reveals the cards of the second row where a $\boxed{\heartsuit}$ card appears in the first row. If there are also $\boxed{\heartsuit}$ cards then the second constraint holds (line must continue straightforward for at least one other dot). If a $\boxed{\clubsuit}$ is revealed then the verifier rejects.

White circle: We now describe how the verifier V can be convinced that the White Circle Rule holds.

1. V forms the same piles as before except that it picks the two consecutive commitments so that each pile has now four cards in it: the first two correspond to the closest commitments of the circle.
2. They do the following enhanced by the input-preserving function evaluation technique.
3. P applies a pile-shifting shuffle to those four piles.
4. V now verifies the constraint that a straight line must pass through a white circle: reveal the first card of each pile. A valid configuration (up to a cyclic shift) is: $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$.
5. Next V picks the two last cards of each pile where a $\boxed{\heartsuit}$ appears previously. Those picked cards correspond to the farthest commitment from the circle in each pile. At this point we have four face-down cards (since two $\boxed{\heartsuit}$s must appear in the previous step).

   V applies the five-card trick using them: If the output is 1, then the line passing through the white circle continues straightforward (as in Fig. 7), and thus, the configuration is not valid. If the output is 0, then the two extremities are perpendicular or only one of them is perpendicular (as in Fig. 6).

Performance evaluation When we have a $p \times q$ board, our protocol uses $4(p+4)(q+4)$ cards in the Input phase, $2(p+4)(q+4)$ cards in the Topology-Preserving Computation phase and 10 cards during the Verification phase (nine numbered cards used for the input-preserving function and one red card for the five-card trick): $6(p+4)(q+4)+10$ cards in total.

7. Security proofs for Masyu

We show that the construction for Masyu satisfies the completeness, extractability, and zero-knowledge properties. As mentioned in the construction, we apply the same topology-preserving computation as the Slitherlink protocol. Hence, the completeness of the Masyu protocol is already proved (see Section 5). It remains to prove the extractability and the zero-knowledge properties.

Extractability The extractability for the protocol holds similarly to our proposed protocol for Slitherlink shown in Section 5. The extractor is exactly the same: it simply reveals all the face-down cards placed after the Topology-preserving computation phase.

Theorem 3. *If the prover does not know a solution for the Masyu puzzle, then the verifier rejects regardless of the prover's behavior.*

To prove Theorem 3, let us show that the resulting loop after the Topology-preserving Computation phase always satisfies the Loop Rule (as in Lemma 1) and any single loop that does not satisfy the other rules is always rejected in the Crossing Verification phase (as in the following Lemma 4). Therefore, any single loop except for the solution is always rejected.

Lemma 4. *Any (single) loop that does not satisfy the Through Circle Rule, Black Circle Rule, and White Circle Rule is always rejected in the Crossing Verification Phase.*

[5] Because of the dummy cards and the sequence cyclically placed, if a circle is on the edge of the board, its next commitment outside the board is a (dummy) 0-commitment.

Proof. Consider any (single) loop that does not satisfy the Through Circle Rule i.e., there are circles where the loop does not pass through. Due to Step 3 in the Crossing Verification Phase, all the left cards of four commitments are turned over (after shuffling them), and hence, the number of 1-commitments is revealed. This means that the verifier can always reject any (single) loop that does not satisfy the Through Circle Rule.

Now, consider any (single) loop that does not satisfy the Black Circle Rule and White Circle Rule, i.e., there are black and white circles where the loop does not respect their respective constraints. Also in Step 3, all the left cards of four closest commitments are turned over (up to cyclic shift), and hence, the perpendicular or straight constraint can be verified. Since the four farthest cards are also turned over, the second part of the constraint in the Black Circle Rule and White Circle Rule are also verified. This notifies the verifier if the loop does not satisfy the Through Circle Rule, Black Circle Rule, and White Circle Rule. □

Proof of Theorem 3. By Lemma 1, the resulting loop after the topology-preserving computation is always single, i.e., it satisfies the Loop Rule. By Lemma 4, if it does not satisfy the remaining rules, the verifier always rejects it in the Crossing Verification phase. That is, any loop except for the solution cannot go through the Crossing Verification phase. □

Zero-knowledge In our construction, all the cards to be opened have been shuffled before being opened. Therefore, all distributions of opened cards can be simulated by a simulator $M(\mathcal{I})$ who does not know the solution. For example, at Step 3 in the Crossing Verification phase, a pile-shifting shuffle has been applied to opened commitments; thus, this is indistinguishable from a simulation putting randomly 1-commitments such that their placement respects the constraint of black circle (perpendicular and then straight line) or white circle (straight line and then at least one perpendicular extremity).

8. Conclusion

We proposed a technique to perform physical ZKP for puzzles that aim to draw a single loop that never crosses itself and never branches off. For this we transform a single loop encoded with physical objects into a new geometrical figure while preserving the single loop. To illustrate our approach, we used this secure computation to construct physical zero-knowledge proof protocols for two Nikoli puzzles: Slitherlink and Masyu.

Of course, our construction has been designed to be used for other puzzles that require a feature of drawing a single loop. For example, *Moon-or-Sun* published by Nikoli aims at drawing a single loop as Slitherlink or Masyu. However, there is an extra rule in this puzzle that are not so easy to take into account in a ZKP protocol. In particular, the following rule is very difficult to deal with: after the loop goes through the moons in one room, it has to go through all the suns in the next room it enters and vice versa.

It should be noted that our latest work [32] for Nurikabe and Hitori was inspired by this study. In that work, we proposed a technique to prove that cells with the same color are connected to each other. In the future work, we aim to deal with famous NP-complete graph theoretic problems such as the Hamiltonian path (and cycle) problem.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported in part by JSPS KAKENHI Grant Numbers JP17K00001, JP19J21153, and JP21K11881. This study was partially supported by the French ANR project ANR-18-CE39-0019 (MobiS5). This work has been partially supported by the French government research program "Investissements d'Avenir" through the IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25) and the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01). This work was also supported by the French ANR project DECRYPT (ANR-18-CE39-0007) and SEVERITAS (ANR-20-CE39-0009).

References

[1] Nikoli Slitherlink, https://www.nikoli.co.jp/en/puzzles/slitherlink.html, 1989.
[2] Y. Abe, M. Iwamoto, K. Ohta, Efficient private PEZ protocols for symmetric functions, in: D. Hofheinz, A. Rosen (Eds.), Theory of Cryptography, in: LNCS, vol. 11891, Springer, Cham, 2019, pp. 372–392.
[3] J. Balogh, J.A. Csirik, Y. Ishai, E. Kushilevitz, Private computation using a PEZ dispenser, Theor. Comput. Sci. 306 (1–3) (2003) 69–84, https://doi.org/10.1016/S0304-3975(03)00210-X.
[4] X. Bultel, J. Dreier, J. Dumas, P. Lafourcade, D. Miyahara, T. Mizuki, A. Nagao, T. Sasaki, K. Shinagawa, H. Sone, Physical zero-knowledge proof for Makaro, in: T. Izumi, P. Kuznetsov (Eds.), SSS 2018, in: LNCS, vol. 11201, Springer, 2018, pp. 111–125.
[5] X. Bultel, J. Dreier, J.-G. Dumas, P. Lafourcade, Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen, in: E.D. Demaine, F. Grandoni (Eds.), Fun with Algorithms 2016, in: LIPIcs, vol. 49, 2016, pp. 8:1–8:20.

[6] Y.-F. Chien, W.-K. Hon, Cryptographic and physical zero-knowledge proof: from Sudoku to nonogram, in: P. Boldi, L. Gargano (Eds.), Fun with Algorithms 2010, in: LNCS, vol. 6099, Springer, 2010, pp. 102–112.
[7] C. Crépeau, J. Kilian, Discreet solitary games, in: D.R. Stinson (Ed.), CRYPTO 1993, in: LNCS, vol. 773, Springer, Berlin, Heidelberg, 1994, pp. 319–330.
[8] P. D'Arco, R.D. Prisco, Secure computation without computers, Theor. Comput. Sci. 651 (2016) 11–36, https://doi.org/10.1016/j.tcs.2016.08.003.
[9] B. den Boer, More efficient match-making and satisfiability: the five card trick, in: J. Quisquater, J. Vandewalle (Eds.), EUROCRYPT 1989, in: LNCS, vol. 434, Springer, 1989, pp. 208–217.
[10] J. Dumas, P. Lafourcade, D. Miyahara, T. Mizuki, T. Sasaki, H. Sone, Interactive physical zero-knowledge proof for Norinori, in: D.-Z. Du, Z. Duan, C. Tian (Eds.), COCOON 2019, in: LNCS, vol. 11653, Springer, 2019, pp. 166–177.
[11] E. Friedman, Pearl puzzles are NP-complete, http://www.stetson.edu/~efriedma/papers/pearl.pdf. (Accessed April 2020).
[12] O. Goldreich, A. Kahan, How to construct constant-round zero-knowledge proof systems for NP, J. Cryptol. 9 (3) (1996) 167–189, https://doi.org/10.1007/BF00208001.
[13] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems, in: STOC 1985, ACM, 1985, pp. 291–304.
[14] R. Gradwohl, M. Naor, B. Pinkas, G.N. Rothblum, Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles, Theory Comput. Syst. 44 (2) (2009) 245–268, https://doi.org/10.1007/s00224-008-9119-9.
[15] R. Ishikawa, E. Chida, T. Mizuki, Efficient card-based protocols for generating a hidden random permutation without fixed points, in: C.S. Calude, M.J. Dinneen (Eds.), UCNC 2015, in: LNCS, vol. 9252, Springer, 2015, pp. 215–226.
[16] J. Kastner, A. Koch, S. Walzer, D. Miyahara, Y. Hayashi, T. Mizuki, H. Sone, The minimum number of cards in practical card-based protocols, in: T. Takagi, T. Peyrin (Eds.), ASIACRYPT 2017, in: LNCS, vol. 10626, Springer, 2017, pp. 126–155.
[17] G. Kendall, A. Parkes, K. Spoerer, A survey of NP-complete puzzles, ICGA J. 31 (2008) 13–34, https://doi.org/10.3233/ICG-2008-31103.
[18] A. Koch, M. Schrempp, M. Kirsten, Card-based cryptography meets formal verification, in: S.D. Galbraith, S. Moriai (Eds.), ASIACRYPT 2019, in: LNCS, vol. 11921, Springer, Cham, 2019, pp. 488–517.
[19] A. Koch, S. Walzer, Foundations for actively secure card-based cryptography, in: M. Farach-Colton, G. Prencipe, R. Uehara (Eds.), Fun with Algorithms 2021, in: LIPIcs, vol. 157, 2020, pp. 17:1–17:23.
[20] A. Koch, S. Walzer, K. Härtel, Card-based cryptographic protocols using a minimal number of cards, in: T. Iwata, J.H. Cheon (Eds.), ASIACRYPT 2015, in: LNCS, vol. 9452, Springer, 2015, pp. 783–807.
[21] J. Kölker, Selected Slither Link variants are NP-complete, J. Inf. Process. 20 (3) (2012) 709–712, https://doi.org/10.2197/ipsjjip.20.709.
[22] P. Lafourcade, D. Miyahara, T. Mizuki, T. Sasaki, H. Sone, A physical ZKP for Slitherlink: how to perform physical topology-preserving computation, in: S. Heng, J. López (Eds.), Information Security Practice and Experience - 15th International Conference, ISPEC 2019, in: LNCS, vol. 11879, Springer, 2019, pp. 135–151.
[23] D. Miyahara, Y. Hayashi, T. Mizuki, H. Sone, Practical card-based implementations of Yao's millionaire protocol, Theor. Comput. Sci. 803 (2020) 207–221, https://doi.org/10.1016/j.tcs.2019.11.005.
[24] D. Miyahara, L. Robert, P. Lafourcade, S. Takeshige, T. Mizuki, K. Shinagawa, A. Nagao, H. Sone, Card-based ZKP protocols for Takuzu and Juosan, in: M. Farach-Colton, G. Prencipe, R. Uehara (Eds.), 10th International Conference on Fun with Algorithms (FUN 2020), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 157, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2020, pp. 20:1–20:21, https://drops.dagstuhl.de/opus/volltexte/2020/12781.
[25] T. Mizuki, Card-based protocols for securely computing the conjunction of multiple variables, Theor. Comput. Sci. 622 (C) (2016) 34–44, https://doi.org/10.1016/j.tcs.2016.01.039.
[26] T. Mizuki, M. Kumamoto, H. Sone, The five-card trick can be done with four cards, in: X. Wang, K. Sako (Eds.), ASIACRYPT 2012, in: LNCS, vol. 7658, Springer, Berlin, Heidelberg, 2012, pp. 598–606.
[27] T. Mizuki, H. Sone, Six-card secure AND and four-card secure XOR, in: X. Deng, J.E. Hopcroft, J. Xue (Eds.), Frontiers in Algorithmics, FAW 2009, in: LNCS, vol. 5598, Springer, 2009, pp. 358–369.
[28] T. Moran, M. Naor, Basing cryptographic protocols on tamper-evident seals, Theor. Comput. Sci. 411 (10) (2010) 1283–1310, https://doi.org/10.1016/j.tcs.2009.10.023.
[29] V. Niemi, A. Renvall, Secure multiparty computations without computers, Theor. Comput. Sci. 191 (1–2) (1998) 173–183, https://doi.org/10.1016/S0304-3975(97)00107-2.
[30] A. Nishimura, Y. Hayashi, T. Mizuki, H. Sone, Pile-shifting scramble for card-based protocols, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 101 (9) (2018) 1494–1502, https://doi.org/10.1587/transfun.E101.A.1494.
[31] L. Robert, D. Miyahara, P. Lafourcade, T. Mizuki, Physical zero-knowledge proof for Suguru puzzle, in: S. Devismes, N. Mittal (Eds.), Stabilization, Safety, and Security of Distributed Systems, in: LNCS, vol. 12514, Springer, Cham, 2020, pp. 235–247.
[32] L. Robert, D. Miyahara, P. Lafourcade, T. Mizuki, Interactive physical ZKP for connectivity: applications to Nurikabe and Hitori, in: L. De Mol, A. Weiermann, F. Manea, D. Fernández-Duque (Eds.), Beyond the Horizon of Computability, in: LNCS, vol. 12813, Springer, Cham, 2021, pp. 373–384.
[33] S. Ruangwises, T. Itoh, Physical zero-knowledge proof for numberlink puzzle and k vertex-disjoint paths problem, New Gener. Comput. 39 (2021) 3–17, https://doi.org/10.1007/s00354-020-00114-y.
[34] S. Ruangwises, T. Itoh, Physical zero-knowledge proof for Ripple Effect, in: S. Hong, S. Nandy, R. Uehara (Eds.), WALCOM: Algorithms and Computation, in: LNCS, vol. 11737, Springer, Cham, 2021, pp. 296–307.
[35] T. Sasaki, D. Miyahara, T. Mizuki, H. Sone, Efficient card-based zero-knowledge proof for Sudoku, Theor. Comput. Sci. 839 (2020) 135–142, https://doi.org/10.1016/j.tcs.2020.05.036.
[36] K. Shinagawa, K. Nuida, A single shuffle is enough for secure card-based computation of any Boolean circuit, Discrete Appl. Math. 289 (2021) 248–261, https://doi.org/10.1016/j.dam.2020.10.013.
[37] A. Stiglic, Computations with a deck of cards, Theor. Comput. Sci. 259 (1–2) (2001) 671–678, https://doi.org/10.1016/S0304-3975(00)00409-6.
[38] K. Takashima, Y. Abe, T. Sasaki, D. Miyahara, K. Shinagawa, T. Mizuki, H. Sone, Card-based protocols for secure ranking computations, Theor. Comput. Sci. 845 (2020) 122–135, https://doi.org/10.1016/j.tcs.2020.09.008.
[39] I. Ueda, D. Miyahara, A. Nishimura, Y. Hayashi, T. Mizuki, H. Sone, Secure implementations of a random bisection cut, Int. J. Inf. Secur. 19 (2020) 445–452, https://doi.org/10.1007/s10207-019-00463-w.
[40] T. Yato, T. Seta, Complexity and completeness of finding another solution and its application to puzzles, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E86-A (5) (2003) 1052–1060, https://search.ieice.org/bin/summary.php?id=e86-a_5_1052.