# A Physical ZKP for Slitherlink: How to Perform Physical Topology-Preserving Computation

Pascal Lafourcade[1][0000−0002−4459−511X], Daiki Miyahara[2,4], Takaaki Mizuki[3][0000−0002−8698−1043], Tatsuya Sasaki[2], and Hideaki Sone[3]

[1] LIMOS, University Clermont Auvergne, CNRS UMR 6158, Clermont-Ferrand, France
[2] Graduate School of Information Sciences, Tohoku University, Sendai, Japan
daiki.miyahara.q4@dc.tohoku.ac.jp
[3] Cyberscience Center, Tohoku University, Sendai, Japan
[4] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

**Abstract.** We propose a new technique to construct physical Zero-Knowledge Proof (ZKP) protocols for games that require a single loop draw feature. This feature appears in Slitherlink, a puzzle by Nikoli. 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, visible to anyone, into the solution loop by using transformations that do not disclose any information about the solution. As a proof of concept, we apply this technique to construct the first physical ZKP protocol for Slitherlink.

**Keywords:** Physical Zero-Knowledge Proof · Slitherlink · Physical Topology Preserving Computation

## 1 Introduction

Zero-Knowledge Proof (ZKP) systems are powerful cryptographic tools that were introduced by Goldwasser, Micali, and Rackoff in [10]. It was then shown that for any NP-complete problem, there exists an interactive ZKP protocol [9]. Later, one of the first physical ZKP protocols was introduced by Naor et al. in [11] for a popular puzzle, Sudoku. In the mentioned article, a prover wants to prove to a verifier that he/she knows the solution of a Sudoku puzzle instance using only physical objects; to this end, in that paper the authors used only cards. Recently in [21], better ZKP protocols have been proposed in terms of numbers of cards used and complexity. They used envelopes and physical tricks to improve the original protocol.

Nikoli[5] 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. In this paper, we focus on *Slitherlink* that was introduced in 1989 in

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

issue the 26th of Nikoli's Puzzle Times. It is also known as *Loop-the-Loop*. It is explained on Nikoli's web site 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)*". Slitherlink was proven to be NP-complete in [23] and other variants in [16]. It means that applying the technique of [9] to construct a ZKP is possible.

Our aim is to propose a physical ZKP protocol for this game. Slitherlink is not like other Nikoli's games since it requires to draw a **single** loop to solve the puzzle. This feature of the game is a challenge that was not present in the previous physical ZKPs for Nikoli's puzzles [4–6, 8, 11, 21].

**Contributions:** We introduce a new 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 in this article but it can be used for any other puzzles that require such type of features in their rules.

**Related works:** Since Naor et al. [11] introduced the first physical ZKP protocol for the Sudoku, physical ZKPs for other puzzles (proven to be NP-complete) have been proposed, e.g., Nonogram [6], Akari, Takuzu, Kakuro, Kenken [4], Makaro [5], and Norinori [8]. All these ZKPs deal with numbers. For example, in Sudoku, a prover has to show the verifier that each column, row, and subgrid contain all the numbers from one to nine.

Physical objects enable us to perform secure computation without relying on computers: such examples are a PEZ dispenser [2], tamper-evident seals [18], and a deck of cards [3]. Among them, secure computation with a deck of cards, called *card-based cryptography*, has been widely studied. Especially, for secure computation of logical AND function, the number of required cards have been reduced in [7, 15, 17, 19, 22], and necessary and sufficient numbers of cards have been provided in [13, 15].

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

**Outline:** In Section 2, we define the rules of Slitherlink, the formal definition of ZKP, and the notation used in this paper. In Section 3, we describe our ZKP protocol for Slitherlink. In Section 4, we show the security proofs of ZKP.

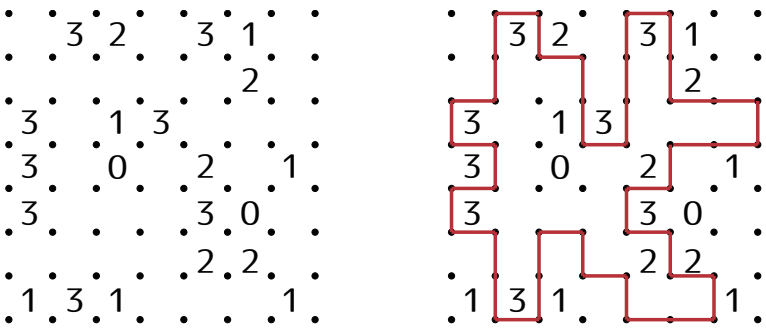


**Fig. 1.** Example of a standard Slitherlink challenge, and its solution.

## 2 Preliminaries

### Rules of Slitherlink

Slitherlink is one of the most famous pencil puzzles published in the puzzle magazine *Nikoli*. 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. Connect vertical/horizontal adjacent dots with lines to make a single loop.
2. Each number indicates the number of lines that should surround its square, while empty squares may be surrounded with any number of lines.
3. The loop never crosses itself and never branches off.

Figure 1 shows an example of a Slitherlink puzzle and its solution; one can easily verify that all conditions are satisfied.

### Zero-Knowledge Proof

A *Zero-Knowledge Proof (ZKP)* is a secure two-party protocol between a prover $P$ and a verifier $V$. Formally, they both have an instance of $\mathcal{I}$ of a problem and only $P$ knows the solution $w$. The prover $P$ wants to convince $V$ that he/she knows $w$ without revealing any information about $w$. Such a proof is called a *zero-knowledge proof*, if it satisfies 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 $V$.

**Zero-Knowledge.** $V$ cannot obtain any information about $w$. Assuming a probabilistic polynomial time algorithm $M(\mathcal{I})$ not containing $w$ if outputs of the protocol and $M(\mathcal{I})$ follow the same probability distribution, the zero-knowledge property is satisfied.

**Notations**

We use the following physical cards: $\boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\heartsuit}\boxed{\heartsuit}$; the black $\boxed{\clubsuit}$ and red $\boxed{\heartsuit}$ cards are called *binary cards*. The backs of all cards are identical and denoted by $\boxed{?}$ . In our construction, binary cards are used to encode the existence of a line while number cards are used for rearranging the positions of cards, as shown later.

*Encoding:* We encode Boolean values with two binary cards as follows: $\boxed{\clubsuit}\boxed{\heartsuit} = 0$ and $\boxed{\heartsuit}\boxed{\clubsuit} = 1$. Two face-down cards 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 protocol, 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. With this encoding, we can represent a loop that is made of several lines. Note that given an $x$-commitment for $x \in \{0, 1\}$, swapping the two cards consisting the commitment results in an $\overline{x}$-commitment; thus, negation can be easily done.

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

*Pile-Shifting Shuffle:* The goal of this operation, which is also called Pile-Shifting Scramble [20], is to *cyclically* shuffle piles of cards. That is, given $m$ piles, each of which consists of the same number of face-down cards, denoted by $(pile_1, pile_2, \ldots, pile_m)$, applying a Pile-Shifting Shuffle results in $(pile_{s+1}, pile_{s+2}, \ldots, pile_{s+m})$:

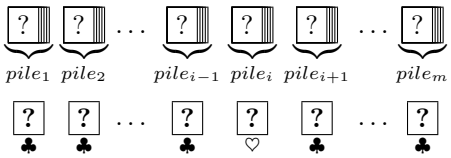
$$\underbrace{\boxed{?}}_{pile_1}\underbrace{\boxed{?}}_{pile_2}\cdots\underbrace{\boxed{?}}_{pile_m} \rightarrow \underbrace{\boxed{?}}_{pile_{s+1}}\underbrace{\boxed{?}}_{pile_{s+2}}\cdots\underbrace{\boxed{?}}_{pile_{s+m}},$$

where $s$ is uniformly and randomly chosen from $\mathbb{Z}/m\mathbb{Z}$. To implement Pile-Shifting Shuffle, we use physical cases that can store a pile of cards, such as boxes and envelopes; a player (or players) cyclically shuffle them by hand until nobody traces the offset. It can be done by physical object as the one created for the physical ZKP for Sudoku in [21].

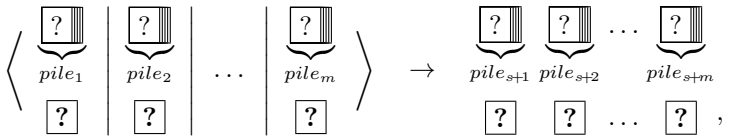
*Pile-Scramble Shuffle:* Pile-Scramble Shuffle is a well-known shuffle operation which was first used in [12]. As mentioned above, let us denote $m$ piles by $(pile_1, pile_2, \ldots, pile_m)$. For such a sequence of piles, applying a Pile-Scramble Shuffle results in $(pile_{r^{-1}(1)}, pile_{r^{-1}(2)}, \ldots, pile_{r^{-1}(m)})$, where $r \in S_m$ is a uniformly distributed random permutation. A Pile-Scramble Shuffle uses similar material as Pile-Shifting Shuffle but its operation is similar to Shuffle.

*Chosen Pile Cut:* It was proposed in [14]. *Chosen Pile Cut* enables a prover to choose a pile $pile_i$ from $m$ piles $(pile_1, pile_2, \ldots, pile_m)$ 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 $m$ cards with their faces down below the piles such that only the $i$-th card is $\boxed{\heartsuit}$:




2. Regarding the cards in the same row as a pile, apply Pile-Shifting Shuffle to the piles (denoted by $\langle \cdot | \ldots | \cdot \rangle$):




   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 $\boxed{\heartsuit}$ appears, and the pile above the revealed $\boxed{\heartsuit}$ is $pile_i$, and hence, we can obtain the desired $pile_i$.

Owing to the Pile-Shifting Shuffle in Step 2, revealing cards leaks no information about $i$ and thus, Chosen Pile Cut leaks no information about $i$, the index of the chosen pile.

## 3 Zero-Knowledge Proof for Slitherlink

In this section, we construct our physical zero-knowledge proof protocol for Slitherlink. 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 Rules 1 and 3 of Slitherlink without the disclosure of any information about the shape.

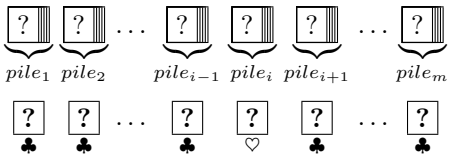
**Verification Phase:** $V$ verifies that the placement of 1-commitments satisfies Rule 2 of Slitherlink.

We introduce some subprotocols in Section 3.1 before presenting our protocol in Section 3.2.

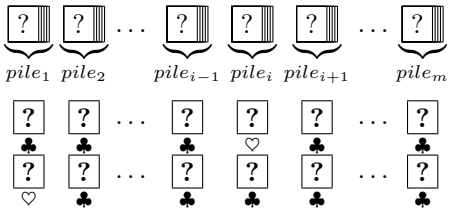
### 3.1 Subprotocols

**Chosen Pile Protocol:** This is an extended version of the Chosen Pile Cut [14] explained in Section 2. Given $m$ piles with $2m$ additional cards, this protocol enables $P$ to choose the $i$-th pile and regenerate the original sequence of $m$ piles.

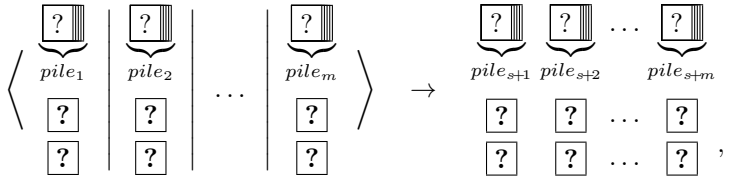
1. Using $m-1$ $\boxed{\clubsuit}$s and one $\boxed{\heartsuit}$, the prover $P$ places $m$ cards with their faces down below the given piles such that only the $i$-th card is $\boxed{\heartsuit}$:



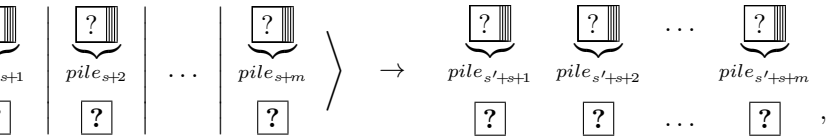

   We further put $m$ cards below the cards such that only the first card is $\boxed{\heartsuit}$:




2. Considering the cards in the same row as a pile, apply a Pile-Shifting Shuffle to the sequence of piles:




   where $s$ is generated uniformly at random from $\mathbb{Z}/m\mathbb{Z}$.
3. Reveal all the cards in the second row. Then, one $\boxed{\heartsuit}$ appears, and the pile above the revealed $\boxed{\heartsuit}$ is the $i$-th pile (and hence, $P$ can obtain $pile_i$). When this protocol is invoked, certain operations are applied to the chosen pile. Then, the chosen pile is placed back to the $i$-th position 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 $\boxed{\heartsuit}$ appears, and the pile above the revealed $\boxed{\heartsuit}$ is $pile_1$. Therefore, by shifting the sequence of piles, we can obtain a sequence of piles whose order is the same as the original one without revealing any information about the order of input sequence.

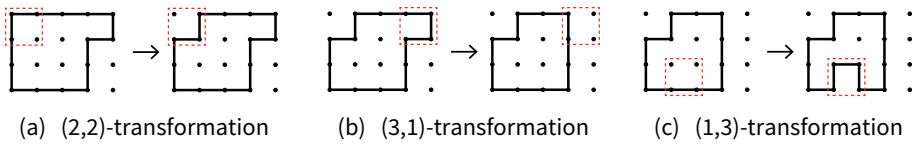
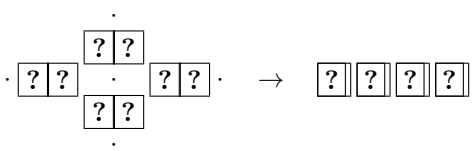



**Fig. 2.** Three transformations.

**Verifying-Degree Protocol:** This protocol can verify that the "degree" of a target vertex (dot) is not four. Here, *degree* means the number of 1-commitments placed around a target vertex. Thus, the prover $P$ wants to prove that there is at least one 0-commitment around the target vertex (when only $P$ knows what the four commitments around the target are).

The Verifying-Degree Protocol proceeds as follows.

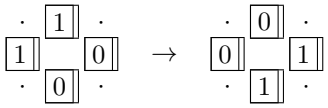
1. Given four commitments that are placed around the target vertex, these can be regarded as a sequence of 4 commitments:



2. By using Chosen Pile Protocol, $P$ chooses one of the 0-commitments. Open the chosen pile to show that it is 0. Now, $V$ is convinced that the degree of the target vertex is not four. Then, $V$ turns over all the opened cards. Because only a 0-commitment is always opened, no information about the four commitments is disclosed.
3. $V$ performs the remaining steps in the Chosen Pile Protocol. Then, all the cards are placed back to their original positions.

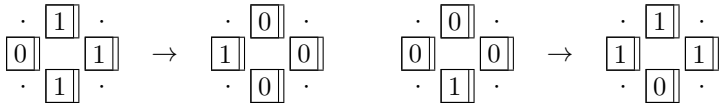
**Topology-Preserving Computation:** This protocol changes a given loop into another loop by one of the three transformations given in Figure 2. Each transformation changes the lines surrounding a square, represented by dash line in Figure 2.

Remember that a line is expressed by a commitment (i.e., two face-down binary cards) in our protocol. Therefore, for example, a (2,2)-transformation means



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

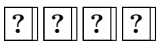
two cards of each commitment:



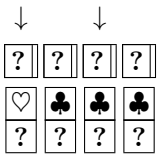
Now, $P$ wants to apply one of the three transformations while the applied transformation is hidden from $V$. Furthermore, $P$ needs to show that the commitments around a target square are "transformable." Note that the three transformations are applicable to four commitments around a square if and only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

Topology-Preserving Computation proceeds as follows.

1. Pick four commitments around a target square:



2. $P$ chooses a 0-commitment facing a 1-commitment using Chosen Pile Protocol.
3. $V$ reveals the chosen commitment and 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.
4. After turning over all the opened cards, $V$ performs the remaining steps in the Chosen Pile Protocol to place all the cards back to their original positions.
5. Swap the two cards of each of the four commitments. (Remember that this results in negating all the four commitments, and hence, a transformation is applied.)
6. $V$ applies a Verifying-Degree Protocol to each of the four dots of the target square. Then, $V$ is convinced that no dots of degree four have been obtained as the result of transformation. This guarantees that the loop was not split and thus, it remains a single loop.

### 3.2 Our Construction

As mentioned at the beginning of this section, the main idea behind our protocol is that the verifier $V$ first creates a big loop and then the prover $P$ transforms the loop into the solution loop one by one. Let us consider a puzzle instance shown in Figure 3 as an example. Our protocol transforms the loop as illustrated in Figure 4.

We are now ready to present the full description of our zero-knowledge proof protocol for Slitherlink.

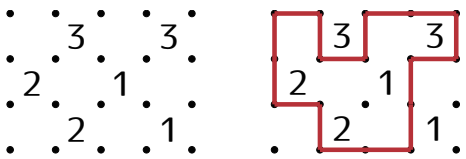



**Fig. 3.** Small example of Slitherlink challenge, and its solution.

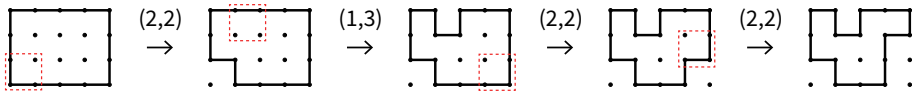
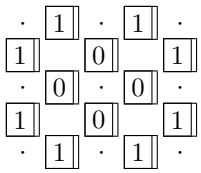


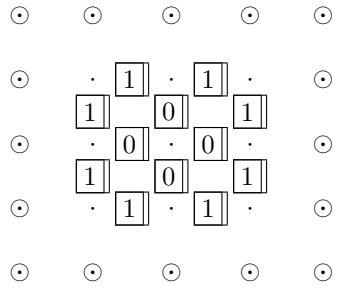

**Fig. 4.** Transformation process.

**Input Phase:** 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 $(2 \times 2)$-square puzzle board:



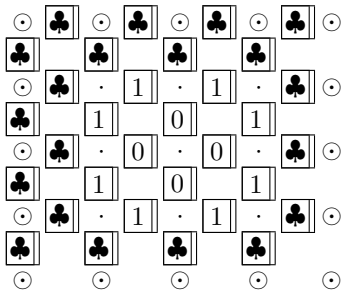

$P$ will apply Topology-Preserving Computation to these commitments to transform the shape of the loop into the solution. Here, $P$ needs to hide the target square. Therefore, we make a sequence of piles from the placed cards, pick the four target commitments using the Chosen Pile Protocol, and apply Topology-Preserving Computation. To properly pick the four commitments, a sequence of piles is formed, as follows.

We first expand the puzzle board by adding dots around the original board. (For explanation, the expanded dots are denoted by $\odot$.)

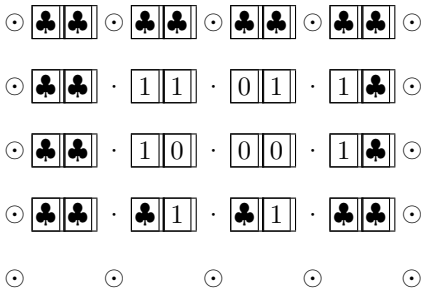



Note that the expanded area is unrelated to the actual puzzle board. $V$ puts dummy commitments on the gaps at the expanded area other than the right and the bottom ends. Each dummy commitment consists of two black cards

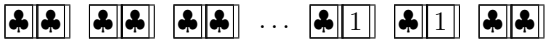
[♣][♣] to prevent the loop from spreading over the expanded area. We denote the dummy commitment by [♣].



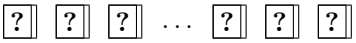
Next, $V$ makes a sequence of 4-card piles as follows. For each square, $V$ first makes a pile from the commitments placed on the left and the top (the commitment on the gap between each vertically consecutive dots is placed on the commitment on its upper right.)



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



to make a sequence of piles:

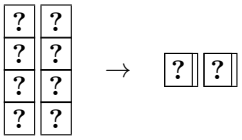


**Topology-preserving computation phase:** In this phase, $P$ applies transformations (explained in Section 3.1) to stepwise change the big loop to the solution loop. Let $n$ be the size of the puzzle instance, namely the number of squares on the puzzle board. Then, note that $P$ can make the solution loop by at most $n$ transformations.

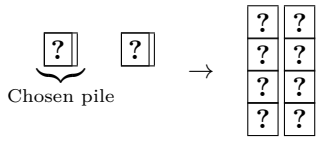
1. $P$ applies the following exactly $n-1$ times such that either the resulting loop is already the solution, or one more transformation will end up the solution. (This is possible because successive two transformations (of the same) to the same square keep the loop unchanged.)

(a) $P$ applies the Chosen Pile Protocol to the sequence of 4-card piles: $P$ picks a 4-card pile composed of left and top edges of the square that $P$ wants to transform. The other edges can be picked by counting the distance from the chosen pile[6].
(b) $P$ applies the Topology-Preserving Computation to the four picked commitments.
(c) $V$ performs the remaining steps in the Chosen Pile Protocol to place the cards back to their original positions.

2. $P$ applies one more transformation or does not change the solution loop so that $V$ does not learn which action occurs, as follows.
(a) Similarly to Step 1 (a) above, $P$ picks four commitments around the target square.
(b) By using the method explained in Topology-Preserving Computation, $V$ confirms that any transformation is applicable.
(c) $V$ arranges the four commitments vertically and makes a pile from each column:



Note that swapping two piles results in inverted value of 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 at left:




(f) $V$ makes four commitments from each row, performs the remaining steps in the Chosen Pile Protocol, and places each commitment back to their original position.

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

**Verification phase:** $V$ is now convinced that the placement of 1-commitments is a single loop (Rule 1) and it never branches off (Rule 3). Therefore, $V$ only needs to verify that the placement satisfies Rule 2 of Slitherlink.

Now, $V$ verifies that the number on each square is equal to the number of lines surrounding it. The verification proceeds as follows, where we virtually

[6] In the above example, the bottom edge corresponds to the pile which is 4 piles away from the chosen pile. Note that the distance between any two piles never changes because only Pile-Shifting Shuffle is applied.

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 left cards (if the square is virtually blue) or all right cards (if the square is yellow) of four commitments around a square on which a number is written:
$$\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$
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 $\boxed{\heartsuit}$ is equal to the number on the square.
   - If $V$ picked all the right cards of four commitments in Step 1, $V$ checks that the number of black cards $\boxed{\clubsuit}$ is equal to the number on the square.
4. Apply Steps 1 to 3 to all other numbered squares. (Note that a commitment is related to at most one blue numbered square and one yellow numbered square.)

Our protocol uses $6(p+2)(q+2)+8$ cards in total, where we have a $p \times q$ board.

## 4 Security Proofs for Our Construction

In this section, we show that our construction satisfies the completeness, extractability, and zero-knowledge properties.

**Completeness:** In the input phase, $V$ is convinced that 1-commitments are placed in a single loop because $V$ does the operations by himself/herself, and hence, $V$ is convinced that the placement satisfies Rules 1 and 3 of Slitherlink. As explained in Section 3.1, the transformations are applied to only applicable squares. Thus, every transformation is performed while preserving Rules 1 and 3. By verifying that the placement satisfies Rule 2 in verification phase, $V$ is convinced that $P$ knows the solution. Therefore, if $P$ has a solution for the puzzle then $P$ can always convince $V$.

Remember 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.*

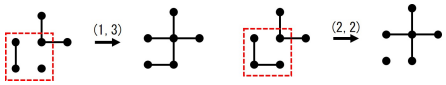
To prove Theorem 1, we first show Lemmas 1 and 2.

**Lemma 1.** *The resulting placement of 1-commitments after the Topology-Preserving Computation always represents a single loop.*

*Proof.* Remember Steps 2 and 6 in the Topology-Preserving Computation: Due to Step 2, the target square is guaranteed to be none of the following two ones (up to rotations).

That is, one of (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 6, the following two transformations that make a loop split cannot occur.

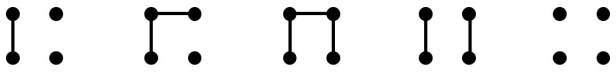



Therefore, it remains a single loop. □

**Lemma 2.** *For any single loop, there is always a (3,1), (1,3), or (2,2)-transformation that increases the number of squares inside the loop by exactly one.*

*Proof.* Consider a single loop; let $\ell$ 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 a (3,1), (1,3), or (2,2)-transformation can be applied to the square such that $\ell$ increases. Note that the loop remains single after the application of the transformation by Lemma 1.

If $\ell \leq 2$, a (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 (a), (b), and (c) exists, all squares outside the loop are either (d) or (e), and hence, it would not be a sigle loop. Therefore, at least one square of type (a), (b), or (c) must exist outside the loop.

Applying a (3,1), (1,3), or (2,2)-transformation to such an external square results in increasing $\ell$ by one. □

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 (3,1), (1,3), or (2,2)-transformation exactly $n - k$ times. □

**Extractability:** Only the person who knows the solution can transform the loop so that the shape satisfies Rule 2. Therefore, $V$ can detect any illegal prover in Verification Phase. Thus, if the prover does not know the solution for a puzzle, then $V$ will be never convinced, irrespective of $P$'s behavior.

More formally, to prove the extractability, we are required to show that any shape that does not satisfy Rule 1, 2, or 3 is always rejected during the protocol.

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

To prove Theorem 2, we show that the resulting loop after the Topology-Preserving Computation always satisfies Rules 1 and 3 (as in Lemma 1) and any single loop that does not satisfy Rule 2 is always rejected in Verification Phase (as in Lemma 3). Therefore, any single loop except for the solution is always rejected.

**Lemma 3.** *Any (single) loop that does not satisfy Rule 2 is always rejected in Verification Phase.*

*Proof.* Consider any (single) loop that does not satisfy Rule 2, 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 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 Rule 2. □

*Proof of Theorem 2.* By Lemma 1, the resulting loop after the Topology-Preserving Computation is always single, i.e., it satisfies Rules 1 and 3. By Lemma 3, if it does not satisfy Rule 2, the verifier always rejects it in Verification Phase. That is, any loop except for the solution cannot go through Verification Phase. □

**Zero-Knowledge:** In our construction, all the opened cards 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 Verification Phase, the Pile-Scramble Shuffle have been applied to 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 square.

## 5 Conclusion

In this study, we introduced a new technique that can transform a single loop encoded with physical objects into a new geometrical figure while preserving the single loop. Furthermore, by using this secure computation, we constructed the first physical zero-knowledge proof protocol for Slitherlink.

As we mentioned in Section 1, our construction can be used for other puzzles that require a feature of drawing a single loop. For example, *Masyu* published

by Nikoli has the same rule as Slitherlink, i.e., we should draw a single loop that never crosses itself and never branches off. Therefore, we can easily construct a physical ZKP protocol for Masyu by executing Topology-Preserving Computation (and then verifying the other rules).

Because physical ZKP protocols should be executed by humans' hands, we usually consider the size of puzzle instance to be bounded by a constant. This differs from conventional ZKP protocols (relying on computers). Note that people enjoying a pencil puzzle will not use a computer to solve it, and hence, physical ZKP protocols are useful and effective for ordinally people.

## Acknowledgement

We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported by JSPS KAKENHI Grant Number JP17K00001 and JP19J21153.

## References

1. http://www.nikoli.co.jp/en/puzzles/slitherlink.html, Nikoli. Slitherlink.
2. Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. Theor. Comput. Sci. **306**(1-3), 69–84 (2003)
3. den Boer, B.: More efficient match-making and satisfiability: *The Five Card Trick*. In: Quisquater, J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 208–217. Springer (1989)
4. Bultel, X., Dreier, J., Dumas, J.G., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: Demaine, E.D., Grandoni, F. (eds.) FUN 2016. LIPIcs, vol. 49, pp. 8:1–8:20 (2016)
5. Bultel, X., Dreier, J., Dumas, J., Lafourcade, P., Miyahara, D., Mizuki, T., Nagao, A., Sasaki, T., Shinagawa, K., Sone, H.: Physical zero-knowledge proof for makaro. In: Izumi, T., Kuznetsov, P. (eds.) SSS 2018. LNCS, vol. 11201, pp. 111–125. Springer (2018)
6. Chien, Y.F., Hon, W.K.: Cryptographic and physical zero-knowledge proof: From sudoku to nonogram. In: Boldi, P., Gargano, L. (eds.) Fun with Algorithms 2010. LNCS, vol. 6099, pp. 102–112. Springer (2010)
7. Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) CRYPTO 1993. pp. 319–330. Springer Berlin Heidelberg, Berlin, Heidelberg (1994)
8. Dumas, J., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: COCOON 2019. LNCS, vol. 11653, pp. 166–177. Springer (2019)
9. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. Journal of Cryptology **9**(3), 167–189 (1991)
10. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: STOC 1985. pp. 291–304. ACM (1985)
11. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. In: Crescenzi, P., Prencipe, G., Pucci, G. (eds.) Fun with Algorithms 2007. LNCS, vol. 4475, pp. 166–182. Springer (2007)

12. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Calude, C.S., Dinneen, M.J. (eds.) UCNC 2015. LNCS, vol. 9252, pp. 215–226. Springer (2015)
13. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 126–155. Springer (2017)
14. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. IACR Cryptology ePrint Archive **2017**, 423 (2017)
15. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer (2015)
16. Kölker, J.: Selected Slither Link variants are NP-complete. Journal of Information Processing **20**(3), 709–712 (2012)
17. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009. Proceedings. LNCS, vol. 5598, pp. 358–369. Springer (2009)
18. Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 285–297. Springer (2005)
19. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theor. Comput. Sci. **191**(1-2), 173–183 (1998)
20. Nishimura, A., Hayashi, Y.i., Mizuki, T., Sone, H.: Pile-shifting scramble for card-based protocols. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **101**(9), 1494–1502 (2018)
21. Sasaki, T., Mizuki, T., Sone, H.: Card-based zero-knowledge proof for Sudoku. In: Ito, H., Leonardi, S., Pagli, L., Prencipe, G. (eds.) Fun with Algorithms 2018. LIPIcs, vol. 100, pp. 29:1–29:10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)
22. Stiglic, A.: Computations with a deck of cards. Theor. Comput. Sci. **259**(1-2), 671–678 (2001)
23. Yato, T., Seta, T.: Complexity and completeness of finding another solution and its application to puzzles. IEICE Trans Fundam Electron Commun Comput Sci (Inst Electron Inf Commun Eng) **E86-A**(5), 1052–1060 (2003)