Balance-Based ZKP Protocols for Pencil-and-Paper Puzzles

Shohei Kaneko, Pascal Lafourcade, Lola-Baie Mallordy, Daiki Miyahara, Maxime Puys, Kazuo Sakiyama

Balance-Based ZKP Protocols for Pencil-and-Paper Puzzles

Shohei Kaneko[1], Pascal Lafourcade[2], Lola-Baie Mallordy[3], Daiki Miyahara[1,4], Maxime Puys[2], and Kazuo Sakiyama[1]

[1] The University of Electro-Communications, Tokyo, Japan
[2] Université Clermont Auvergne, CNRS, Clermont Auvergne INP, Mines Saint-Etienne, LIMOS, 63000 Clermont-Ferrand, France
[3] University Clermont Auvergne, LIMOS, CNRS UMR (6158), Aubière, France
[4] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract. In this paper, we propose zero-knowledge proof (ZKP) protocols using physical objects for four pencil-and-paper puzzles: the well-known Sudoku as well as Makaro, Futoshiki, and Kakuro. That is, our protocols allow a prover to convince a verifier that the prover knows a solution to a puzzle without relying on the use of computers. While previous physical ZKP protocols for puzzles have mainly relied on decks of cards, our research introduces a novel approach utilizing a balance scale and coins to design balance-based ZKP protocols; moreover we show its flexibility by adapting it to the four different puzzles. We compare the number of coins and operations in our protocols with the existing card-based protocols and show that, for certain puzzles, our balance-based protocol outperforms the card-based method. Finally, we prove that our protocols achieve perfect completeness, perfect soundness and are perfectly zero-knowledge.

Keywords: Card-based cryptography · Zero-knowledge proof · Balance scale · Pencil-and-paper puzzle · Sudoku.

1 Introduction

Alice, a hungry girl, goes to a fish market in the East Coast of the US with no money as depicted in Fig. 1. She spots a stand selling fish, with a big sign claiming *"Free fish for anyone who can solve my four puzzles"*. She comes closer and sees that the puzzles are four pencil games: Sudoku, Makaro, Futoshiki, and Kakuro. She cannot miss such a golden opportunity, and starts searching for the solutions. After several hours racking her brain without finding any solution, she screams at the merchant: *"Grifter, your puzzles are impossible!"*. The merchant calmly tells her *"I can prove I know all of the solutions"*. The merchant cannot give away its solutions, or people would come flocking to its stand asking for free fish. What he needs to do is a *zero-knowledge proof* (ZKP) to Alice, allowing to convince her that he knows a solution without revealing it. He remembers that Murata et al. [**?**] proposed similar protocols using a PEZ dispenser. However,

there are no PEZ dispensers at the fish market; the merchant only has a *balance* and *coins* on his stand, to weigh the fishes he sells. In this research, we propose a method to assist merchants, designing ZKP protocols using a balance and coins.

1.1 Zero-Knowledge Proof

Zero-knowledge proofs (ZKPs), introduced in 1985 by Goldwasser et al. [9], allow a prover P to convince a verifier V that a given statement is true without revealing any further information. ZKPs give a model that is not limited to computer use, but may also be applied in real life using everyday objects. In 1990, Quisquater et al. [22] published the well-known story of the Ali Baba cave to illustrate this concept, which made the first instance of a physical ZKP.

A ZKP protocol for a solution to a pencil-and-paper puzzle should satisfy three properties as follows:

Completeness: If P knows a solution of a given grid, it can convince V.

Soundness: If P does not provide a correct solution of a given grid, V rejects P with a sufficiently high probability.

Zero-knowledge: The verifier V is not given any information other than that the prover P can solve the puzzle.

These properties can come in three different flavours: perfect, statistical and computational. Perfect completeness means that an honest prover will always convince an honest verifier on a true statement, perfect soundness means that it is impossible to prove a false statement, and perfect zero-knowledge means that transcripts can be perfectly simulated and leak no information whatsoever. Perfect soundness can be relaxed to statistical soundness, where a prover must have a negligible probability of falsely convincing the verifier. It can be relaxed further to computational soundness, where any way to cheat must be computationally infeasible. Completeness and zero-knowledge can be relaxed in the same way. Our proposed protocols achieve the stronger versions of these properties: perfect completeness, perfect soundness, and perfect zero-knowledge based on some physical assumptions.

It was shown that for any NP-complete problem, there exists an interactive ZKP [8]. An extension by Ben-Or et al. [3] showed that every provable statement can be proven in zero-knowledge. The puzzles introduced in this paper have all been proven to be NP-complete: Sudoku and Kakuro in 2003 [32], Makaro in 2018 [14], and Futoshiki in 2021 [16]. Thus, there should exist ZKP protocols

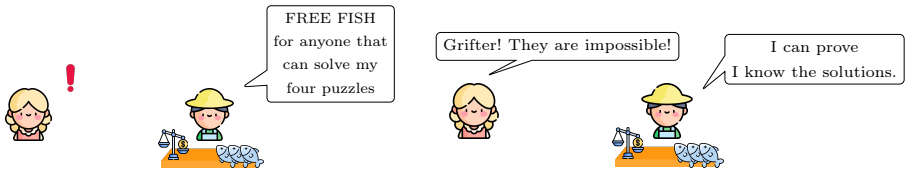

Fig. 1: Alice visits the fish market.

Table 1: Comparison of the complexity of our balance-based protocols with the existing card-based ones. We consider $n \times n$ grids except for the standard 9×9 Sudoku grid. In Sudoku, n_e is the number of empty cells. In Futoshiki, i is the number of inequality symbols. In Makaro, n_s represents the number of rooms with size s, $2 \leq s \leq s_0$ for some s_0, c is the number of arrow cells, d is the number of cells adjacent to arrow cells, and e is the number of bold lines between two adjacent cells in different rooms. In Kakuro, t represents the number of triangular cells, n_h is the number of uninterrupted rows and columns of length h, $2 \leq h \leq h_0 < n$ for some h_0, and w is the number of white cells.

	Balance-Based			Card-Based	
	Coins	Shuffles	Comparisons	Cards	Shuffles
Sudoku	243	27	$686 + 2n_e$	90 [28]	45 [28]
Futoshiki	n^2	$2n$	$2(n-1)(\sum_{k=1}^{n} k) + 2\binom{n}{2} + n_e + i$	–	–
Makaro	$\sum_{s=2}^{s_0} sn_s + 2e + s_0$	$\sum_{s=2}^{s_0} n_s + e$	$\sum_{s=2}^{s_0}(n_s \times \sum_{k=1}^{s} k) + d - c + 3e$	$2s_0 - 1 + s_0 \sum_{s=2}^{s_0} n_s) + (s_0 - 1)s_0$ [5]	$2(\sum_{s=2}^{s_0} n_s + c + e)$ [5]
Kakuro	$t + 2w$	t	$t + \sum_{\ell=2}^{\ell_0}(n_\ell \binom{\ell}{2}) + w$	$81(t + 81)$ [19]	$3t + 1$ [19]

for such puzzles; however, a concrete procedure using a balance has not been addressed.

1.2 Contributions

We propose a new perspective on ZKPs for pencil puzzles, replacing decks of cards with a balance and coins. To prove our method's adaptability, we show it can be applied to four different puzzles. We develop ZKP protocols for Sudoku, Makaro, Futoshiki, and Kakuro, which all provide perfect completeness, perfect soundness, and perfect zero-knowledge.

Table 1 indicates the number of coins, shuffles, and comparisons used in our balance-based protocol, as well as the number of cards and shuffles used in the existing card-based protocols. In Kakuro, it can be observed that the number of coins used in the balance-based protocol is less than the number of cards used in the card-based protocol [19]. In Futoshiki, our balance-based protocol directly verifies an inequality using the property of a balance, although there is no card-based protocol yet. From these observations, it can be inferred that in certain puzzles, balance-based protocols may reduce the number of physical entities and rounds of operation, making them easier to execute compared to card-based protocols. We note that a type of shuffle used in card-based ZKP protocols is costly to implement, while our balance-based protocol uses a common and easy-to-implement shuffle.

1.3 Related Work

In 2009, the first physical ZKP applied to Sudoku was proposed [10], using a deck of cards. This leads to several improvement results [26, 28, 30]. In addition to Sudoku, there are many card-based ZKP protocols, such as Nurimisaki [25], Kurodoko [25], Juosan [18], Usowan [?], Herugolf [?], Five cells [?, ?], Nurikabe [23], Hitori [23], Heyawake [23], Makaro [5, 26], Nonogram [?, ?], Numberlink [?], Bridges [?], Cryptarithmetic [?], Akari [4], Takuzu [4, 18], Kakuro [4, 19], KenKen [4], Shikaku [?], Slitherlink [?], Sumplete [?], Suguru [?], Topswops [?], Pancake Sorting [?], ABC End View [?, ?], Ball Sort [?], Goishi Hiroi [?], Ripple Effect [27], 15 puzzle [?], graph problems [?], and Moon-or-Sun [11]. However, few solutions try to incorporate other everyday objects other than cards. Such examples include: PEZ dispenser [?, 1, 2], coins (where their weights are not considered, but rather the toss of coins with result either head or tail) [?, 15], polarizing plates [29], dial lock [20], tamper-evident seals [21], balls in bags [17], and marbles in an auction protocol [6]. These protocols are not for ZKPs, but for secure multiparty computations, which enable us to compute a given function over private inputs without revealing anything. Our study proposes a ZKP protocol using a balance and coins, which to our knowledge is the first of its kind. That is, our study explores computations performed by comparisons. It should be noted that a ZKP protocol for Sudoku can be constructed by comparisons. As for Futoshiki, we are the first ones to propose a physical ZKP protocol. Our ZKP protocols using everyday objects could inspire further research. This study, along with research on card-based cryptography, aims to answer the question: "What is the easiest way to perform cryptographic tasks, such as secure computation and zero-knowledge proof systems, without using computers?"

The computational complexity of pencil-and-paper puzzles has been widely studied [13], and a large number of puzzles are proved to be NP-complete as shown in a recent survey [31]. To the best of our knowledge, no research has employed an NP-hardness proof to construct a physical ZKP protocol because such an NP-hardness proof is shown by a reduction (mostly from the SAT problem), while a physical ZKP protocol is constructed directly to eliminate overhead.

2 Model

To describe our ZKP protocols visually, we introduce our use of coins and balances. We use balances similar to the Roberval balance, which is depicted in Fig. 2. We assume an ideal balance that tilts at a constant angle toward the heavier side regardless of the weight difference. That is, from seeing how the balance tilts, we cannot obtain information on the weights of coins placed on its plates other than which is heavier.

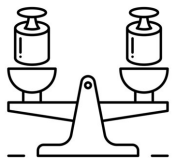

Fig. 2: The Roberval balance.

2.1 Actions

Our protocols use a coin represented as ◯ of various weights (their weights are indistinguishable by just looking at them). We sometimes denote a stack of several coins by ◯◯◯ for simplicity. In addition to moving coins, our protocols make use of two specific operations on coins: *Compare* and *Shuffle*.

Compare: We represent a comparison of two stacks of coins using a balance as follows: ◯◯◯ | ◯◯◯. This operation returns three results: *Left*, *Right*, and *Even*, referring to the heavier stack between the two stacks, and no more information.

Shuffle: Shuffling several coins is represented as follows: $[\bigcirc_1 \bigcirc_2 \cdots \bigcirc_m] \rightarrow [\bigcirc_{r(1)} \bigcirc_{r(2)} \cdots \bigcirc_{r(m)}]$, where r is a uniformly distributed random permutation chosen from the symmetric group of degree m. This operation returns the coins rearranged completely random: after shuffling, the order of m coins is rearranged according to r (the underscripts number are given to identify the new positions; however, the coins are indistinguishable in practice).

2.2 Protocol

In our balance-based ZKP protocol for a puzzle, a prover P first places a coin on a cell of the given grid, such that the weight of the coin represents the number filled in the cell in a solution P has. This means that P needs to know a specific weight of a coin beforehand without a verifier V knowing it. This is possible if P uses a balance without V observing it, and we omit this in a protocol description. Additionally, only P should handle coins (whose weights directly represent a solution) throughout protocols, so as not to give V any information on the weights of the coins.

3 Sudoku

Sudoku is a famous puzzle, which gained popularity in 1986 when it was published by the Japanese puzzle company, Nikoli[5]. In this game, any number from 1 to 9 is placed in an empty cell. A typical Sudoku grid is a 9×9 grid, divided into 3×3 blocks. Initially, some cells are filled with numbers. In Fig. 3, we give a simple example of a Sudoku grid and its solution. The goal is to fill the cells so that each row (there are 9 rows), each column (there are 9 columns), and each block (there are 9 blocks) contains distinct numbers from 1 to 9.

3.1 ZKP Protocol

We present a ZKP protocol for Sudoku using a balance and coins. That is, this protocol enables a prover P (*i.e.*, the merchant) to convince a verifier V (*i.e.*, the hungry girl) that P knows a solution of a given Sudoku puzzle (in the fish market introduced in Sect. 1). Its security proof is given in Sect. 3.2.

[5] https://www.nikoli.co.jp/en/

8					5	1		
		1				8		
	4		2				9	
				3				2
1	2	3	4		6	7	8	9
6				1				1
	8				9	5		
		2				4		
		7	6					

8	3	9	7	6	5	1	2	4
2	6	1	3	9	4	8	7	5
7	4	5	2	8	1	3	9	6
5	9	4	8	3	7	6	1	2
1	2	3	4	5	6	7	8	9
6	7	8	9	1	2	5	4	3
3	8	6	1	4	9	2	5	7
9	1	2	5	7	3	4	6	8
4	5	7	6	2	8	9	3	1

Fig. 3: An example of a Sudoku puzzle and its solution introduced in the Nikoli's website: https://www.nikoli.co.jp/en/puzzles/sudoku/.

Setup Phase: According to P's solution, the prover P places three ◯s on each cell so that the weight of each coin placed on a cell is equal to the number filling the cell. More precisely, the coins are placed in two phases:

1. For each initially filled cell, V places three ◯s, each with the corresponding weight.
2. For each empty cell, P places three ◯s according to its solution.
3. V checks that P executed the setup honestly, *i.e.*, all the three coins placed on each empty cell have the same weight. For this, P performs two comparisons for each empty cell and confirms that they result in even, as follows:

$$\bigcirc_1 \bigcirc_2 \bigcirc_3 \;\rightarrow\; \bigcirc_1 \mid \bigcirc_2 \;\&\; \bigcirc_1 \mid \bigcirc_3,$$

where the three coins are specified by the subscripts.

Verification Phase: The prover and the verifier execute the following steps to confirm that the nine coins placed in the first row (resp. column or block) match those in each of the other rows (resp. columns or blocks), *i.e.*, the numbers 1 through 9 each appear only once in each row (resp. column or block).

1. P picks a ◯ on each cell of the first row and shuffle them: [◯◯◯◯◯◯◯◯◯] → ◯◯◯◯◯◯◯◯◯. Let us denote these nine coins by $R_1 = \{\bigcirc^1_1, \bigcirc^1_2, \ldots, \bigcirc^1_9\}$.
2. V confirms that R_1 has all different weights, as follows:
   - For each of all possible pairs in R_1, *i.e.*, $\bigcirc^1_i$ and $\bigcirc^1_j$, $1 \le i < j \le 9$, the prover P compares them as follows:

$$\bigcirc^1_i \mid \bigcirc^1_j \quad \text{for all } i \text{ and } j \text{ such that } 1 \le i < j \le 9.$$

   - If the comparison results in even, then V rejects P's solution.
3. P picks a ◯ on each cell of another row, say the k-th row, $2 \le k \le 9$, and shuffle them: [◯◯◯◯◯◯◯◯◯] → ◯◯◯◯◯◯◯◯◯. Let us denote these coins by $R_k = \{\bigcirc^k_1, \bigcirc^k_2, \ldots, \bigcirc^k_9\}$.
4. V confirms that only a single coin among R_1 has the same weight as a coin among R_k. For this, it proceeds as follows, for each coin $\bigcirc^1_i$, $1 \le i \le 9$, among R_1.

(a) P compares $\bigcirc_i^1$ with each coin $\bigcirc_j^k$ among R_k, as follows:

$$\bigcirc_i^1 \mid \bigcirc_j^k \quad \text{for all } \bigcirc_j^k \in R_k.$$

- If the comparison results in even, then P returns $\bigcirc_i^1$ to R_1 but removes $\bigcirc_j^k$ from R_k.
- Otherwise, P returns the two coins.

(b) If none of the above comparisons result in even, V rejects P's solution.

5. Repeat the above steps from step 3 for another k. Note that there is no need to shuffle R_1.
6. Repeat the above steps for columns and blocks.

In this way, the verifier V is convinced that each row (resp. column or block) contains distinct numbers from 1 to 9, because for any two rows (resp. columns or blocks), there exists exactly one cell that contains the same number as the one in the other row (resp. column or block). Note that P is the only one manipulating the coins associated with the solution, otherwise V could learn information on their weights when manipulating them.

Efficiency: The numbers of coins, shuffles, and comparisons are summarized in Table 1. Let n_e denote the number of empty cells in a given Sudoku grid. This protocol uses $243\,(= 3 \times 81)$ coins, $27\,(= 9 \times 3)$ shuffles, and $1188 + 2n_e$ comparisons (but will be improved in Sect. 3.3). Let us count the number of comparisons. First, in the setup phase, two comparisons are performed for each empty cell, *i.e.*, $2n_e$ comparisons. In the verification phase, we first compare nine coins in R_1 one by one, *i.e.,* $36\,(= \binom{9}{2})$ comparisons. Then we compare nine coins in R_1 with nine coins in R_2 one by one, but the number of coins in R_2 decreases by one after comparing a coin in R_1 with all coins in R_2. Thus, in the worst case, we need $45\,(= \sum_{k=1}^{9} k)$ comparisons. Because there are eight rows (resp. columns or blocks) excluding the first row, the number of comparisons becomes $1188\,(= 36 \times 3 + 45 \times 8 \times 3)$. Therefore, the total number of comparisons becomes $1188 + 2n_e$. For an $n \times n$ Sudoku grid, this protocol uses $3n^2$ coins and requires $3n$ shuffles and $3(n-1)\sum_{k=1}^{n} k + 3\binom{n}{2} + 2n_e$ comparisons. Table 1 shows the case for $n = 9$.

3.2 Security

We prove the three properties of ZKP for our proposed protocol.

Lemma 1 (Perfect Completeness – Sudoku). *If P provides a correct solution of a given Sudoku grid, V is always convinced.*

Proof. If P provides a correct solution, it is clear that both R_1 and R_k can be regarded as $\{1, 2, \ldots, 9\}$. Therefore, comparing every pair of numbers in R_1 cannot result in even, and comparing a number in R_1 with each number in R_k should result in even when the same numbers are compared. □

Lemma 2 (Perfect Soundness – Sudoku). *If P does not provide a correct solution of a given Sudoku grid, V always rejects P's solution.*

Proof. Without loss of generality, assume that P gives an incorrect solution for a row, *i.e.*, there are two or more coins of the same weight $\ell \in \{1, \ldots, 9\}$ among nine coins in the same row. If such coins are included in R_1, then they are detected when comparing them because every possible pair of coins in R_1 is compared in step 2. Otherwise, *i.e.*, they are included in any R_k, $2 \leq k \leq 9$, they can be detected in step 4 as follows. The first time that P compares such a coin of weight ℓ in R_k with a coin of weight ℓ in R_1, the coin of weight ℓ is removed from R_k. Then, since R_1 no longer contains a coin of weight ℓ, there should exist a coin in R_1 such that no comparison with it and any coin in R_k results in even. Therefore, V can always rejects an incorrect solution. □

Lemma 3 (Perfect Zero-Knowledge – Sudoku). *The verifier V is not given any information other than that the prover P can solve a given Sudoku grid.*

Proof. Note that V must not know the weight of even one coin placed on an empty cell; otherwise, V knows a number filled with the corresponding cell in the solution. Since the weight of a coin is indistinguishable from its appearance, once P places a coin on a cell, its weight cannot be known unless V picks it. Note that our protocol lets P handle the coins when they need to be moved or touched as noted in Sect. 2.2.

Informally, our protocol is zero-knowledge because it shuffles nine coins in R_1 and R_k in steps 1 and 3, respectively, to hide information about positions of cells where the nine coins are originally placed. This means that comparisons in steps 2 and 4 leak no information about the solution, except whether the placement of coins is valid or not. □

Formally, we construct a simulator not in possession of a solution of a given puzzle to prove the zero-knowledge property. For this, we refer to [10], where a simulation-based proof for a card-based ZKP protocol for Sudoku is provided. Their proof replaces the standard rewind ability for a simulator with the ability to arbitrarily swap sequences of cards with another at any time. We apply this technique to construct a simulator for our protocol, ensuring indistinguishability from an honest prover, *i.e.*, it has the ability to arbitrarily swap coins with another one. The simulator acts as follows.

- In the setup phase, the simulator places three identical coins of any weight on each empty cell.
- In step 1 of the verification phase, when R_1 is shuffled, the simulator swaps R_1 with nine coins of weights 1 through 9.
- In step 3, when R_k is shuffled, the simulator swaps R_k with nine coins of weights 1 through 9.

Since a shuffling action provides a randomized order of coins, the comparison results performed in steps 2 and 4 are completely indistinguishable from those provided by an honest prover. □

3.3 Discussion

Number of Comparisons: We can further reduce the number of comparisons by employing a method similar to the binary search when comparing R_1 with R_2. This is because after executing step 2, the weight of every coin in R_1 is determined from the 36 comparison results performed in step 2. If we compare a coin of weight five with every coin in R_2, then we can classify R_2 into four coins of weights less than five, four coins of weights greater than five, and the coin of weight five. This reduces the number of comparisons required, because the four coins of weights less (greater) than five do not need to be compared with the one of weight greater (less) than five. Recursively applying this method, the number of comparisons required for comparing R_1 and R_2 is 25, which is less than the one derived in the efficiency paragraph, *i.e.*, 45. The total number of comparisons is $686 (= 36 + 25 \times (8 + 9 + 9)$, where R_1 used for the verification of rows is also used for columns and blocks (excluding comparisons performed in the setup phase). Moreover, another improvement can be considered, such as rearranging the order of comparisons, *e.g.*, comparing R_3 and R_4 after comparing R_1 and R_2 and then comparing R_3 and R_1 might reduce the number of comparisons.

Card-based Protocol: Placing three identical coins on each cell is the same as in the existing card-based ZKP protocol proposed by Gradwohl *et al.* [10], in which three cards each having the same number are placed on each cell. Note that this protocol is different from the protocol in Table 1. Their protocol does not confirm that each of three cards has the same number, and hence, it does not achieve perfect soundness. In our protocol, since we encode a number with the weight of a coin, we can confirm that each of three coins has the same weight using a balance, achieving perfect soundness.

4 Futoshiki

Futoshiki is a puzzle developed by Tamaki Seto in 2001, played on an $n \times n$ square grid. A Futoshiki grid includes white cells and inequality signs. In Fig. 4, we give an example of a 4×4 Futoshiki grid and its solution. The goal is to place one number in every white cells on the board according to the following constraints:

1. Each row and each column contains all the numbers 1 through n.
2. The numbers must satisfy the inequality signs.

4.1 ZKP Protocol

The main difference from Sudoku is that the numbers must also satisfy the inequality rule. We detail our protocol for an $n \times n$ grid. Our protocol achieves perfect completeness, perfect soundness and is perfectly zero-knowledge. The proofs are given in Sect. 4.2.

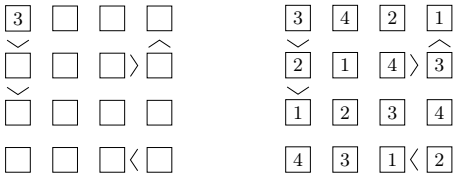

Fig. 4: An example of a Futoshiki puzzle and its solution generated in Futoshiki.com: https://www.futoshiki.com.

Setup: The setup phase is almost the same as in our proposed protocol for Sudoku (Sect. 3). That is, according to P's solution, the prover P places two ◯s on every cell, and P compares the two ◯s to show V that they have the same weight.

Verification: P and V execute the following steps:

1. To verify that the numbers on both sides of each inequality sign satisfy the rule, P compares two ◯s placed on both sides of the sign: ◯ | ◯. V observes that the balance gives the expected result; if not, V rejects P's solution. After performing each comparison, P moves the ◯s to their original positions.
2. To verify that each row and column contains all the numbers from 1 to n, P and V use the same method as for Sudoku (Sect. 3).

Efficiency: Let i denote the number of inequality sings in a given $n \times n$ grid. This protocol uses n^2 coins and performs $2n$ shuffles and $2(n-1)(\sum_{k=1}^{n} k) + 2\binom{n}{2} + n_e + i$ comparisons. Compared to Sudoku, the number of comparisons is reduced by $(n-1)\sum_{k=1}^{n} k + \binom{n}{2}$ due to the absence of blocks, but it increases by i for the inequality verification. The shuffles are also reduced by n compared to Sudoku due to the absence of blocks. In the inequality verification, because no shuffling is performed, it does not impact the total number of shuffle operations.

4.2 Security

We prove the security of the Futoshiki protocol. A proof for completeness is omitted because it is clear from the protocol description.

Lemma 4 (Perfect Completeness – Futoshiki). *If P knows a solution of a given Futoshiki grid, he can always convince V.*

Lemma 5 (Perfect Soundness – Futoshiki). *If P does not provide a correct solution of a given Futoshiki grid, V always rejects P.*

Proof. When P gives an incorrect solution, the following two situations are possible:

– A row (resp. column) contains the same number at least twice. In this case, V will reject P's solution in the same way as in Sudoku (see Lemma 2).

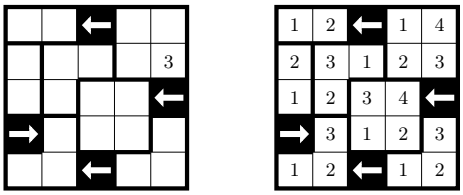

Fig. 5: An example of a Makaro puzzle and its solution introduced in the Nikoli's website: https://www.nikoli.co.jp/en/puzzles/makaro/.

– A pair of numbers does not verify the inequality sign between them. In this case, when V observes the result of the comparison, it will notice that the inequality is not satisfied and V will reject P's solution.

Therefore, V will always reject an invalid solution. □

Lemma 6 (Perfect Zero-Knowledge – Futoshiki). *The verifier V is not given any information other than that the prover P can solve the Futoshiki grid.*

Proof. We show that no information has been leaked other than that the prover P can solve the Futoshiki grid in both of the verification phases:

– In step 1, when V checks whether the numbers satisfy the inequality rule, as V does not touch the coins but only observes the result of the balance, V can only learn which coin is heavier (the coins are visually indistinguishable). Hence V still does not learn anything on P's solution except that it is correct.
– In step 2, V cannot learn anything on the numbers P placed on each cell for the same reason as in the Sudoku ZKP protocol (Lemma 3).

Therefore, V cannot learn anything throughout the whole process, except whether the solution is valid or not.

Formally, a simulator acts as follows.

– In the setup phase, the simulator arbitrarily places two identical coins on each empty cell.
– In step 1, before comparing two coins, it swaps the two coins so that the comparison result matches the inequality sign.
– In step 2 of the verification phase, it acts in the same way as in the proof for Lemma 2.

Since the simulator places coins to respect the inequality signs, the comparison results performed in step 1 are indistinguishable from those provided by an honest prover. □

5 Makaro

Makaro is another grid game proposed by Nikoli. A Makaro grid is made of white cells, and black cells filled with an arrow. In Fig. 5, we give an example of a 5×5

Makaro grid. The goal is to place one number in every white cells on the grid according to the following constraints:

1. The areas separated by bold lines are called rooms, and each room is filled with one number from 1 to the number of cells in that room.
2. In the case of a black cell with an arrow, the cell to which the arrow points must be the cell with the highest number out of the vertically and horizontally adjacent cells to that black cell.
3. Adjacent cells cannot have the same number.

5.1 ZKP Protocol

The main difference from Sudoku is that the cells must be filled according to the arrow rule, *i.e.*, the number pointed by the arrow must be the highest among the adjacent cells. This property is easy to verify using a balance. Our protocol achieves perfect completeness, perfect soundness and is perfectly zero-knowledge. The proofs are given in Sect. 5.2.

Setup: As in the protocol for Sudoku (Sect. 3), according to P's solution, the prover P places several ◯s each of the corresponding weight on every cell, and P compares them to show V that they have the same weight. In the following, we assume that the number of coins placed is sufficient for clarity. The correct number of coins is computed later.

Verification: The prover P and the verifier V execute the following steps:

1. To verify that a ◯ on each cell pointed by an arrow is the heaviest, P compares it with each other adjacent ◯ around the arrow: ◯ | ◯. If the coin pointed by the arrow is ever found lighter than another, V rejects P's solution. The coins are moved to their original positions after each comparison.
2. Let $s_0\,(\geq 2)$ denotes the size of the largest room in the given grid. To verify that each room contains different coins with weight from 1 to the size of the room, it proceeds as follows.
   (a) V prepares additional s_0 coins with weight from 1 to s_0.
   (b) For a room of size s, $1 \leq s \leq s_0$, the prover P picks a ◯ from each cell and shuffle the s coins: [◯◯◯]. Let R denote these s coins.
   (c) Let R_0 denote the s coins with weight from 1 to s among the s_0 coins prepared. Using R_0 and R, the verifier V confirms that they are identical in the same way as for Sudoku (Sect. 3) as follows.
      i. P compares a coin in R_0 with each coin in R.
         – If the comparison results in even, then P returns ◯ to R_0 but removes ◯ from R.
         – Otherwise, P returns the two coins.
      ii. If none of the above comparisons result in even, V rejects P's solution.
      iii. Repeat step 2(c)i for another coin in R_0 until all the coins in R_0 are compared.
   (d) Repeat step 2(b) for another room until all the rooms are confirmed.

Table 2: The numbers required for each verification in our Makaro protocol

	Coins	Shuffles	Comparisons
Setup	–	–	$2e$
Arrow	–	–	$d-c$
Room	$\sum_{s=2}^{s_0} sn_s + s_0$	$\sum_{s=2}^{s_0} n_s$	$\sum_{s=2}^{s_0}(n_s \sum_{k=1}^{s} k)$
Adjacency	$2e$	e	e
Total	$\sum_{s=2}^{s_0} sn_s + s_0 + 2e$	$\sum_{s=2}^{s_0} n_s + e$	$\sum_{s=2}^{s_0}(n_s \sum_{k=1}^{s} k) + d - c + 3e$

3. To verify that no identical coins are next to each other, for each of such a pair of coins, P shuffles the two ◯s: [◯◯], and compares them: ◯ | ◯. If the balance shows even, V rejects P's solution. The two coins are no longer used and are removed.

Efficiency: Let n_s denote the number of rooms with size s, $2 \leq s \leq s_0$ for some s_0, c the number of arrows, d the number of cells adjacent to arrow cells, and e the number of bold lines between two adjacent cells in different rooms. This protocol uses $\sum_{s=2}^{s_0} sn_s + s_0 + 2e$ coins and performs $\sum_{s=2}^{s_0} n_s + e$ shuffles and $\sum_{s=2}^{s_0}(n_s \sum_{k=1}^{s} k) + d - c + 3e$ comparisons. Table 2 summarizes those numbers required for each verification in the protocol. We first note that the number of rooms is represented as $\sum_{s=2}^{s_0} n_s$ and the number of cells is $\sum_{s=2}^{s_0} sn_s$. Therefore, in the room verification, the protocol uses a coin placed on every cell as well as the additional s_0 coins, *i.e.*, $\sum_{s=2}^{s_0} sn_s + s_0$ coins, and shuffles coins placed on each room, *i.e.*, $\sum_{s=2}^{s_0} n_s$ shuffles. The number of comparisons follows the same approach as for Sudoku: for a room of size s, the number of comparisons is $\sum_{k=1}^{s} k$ in the worst case, and since there are n_s rooms of size s for $2 \leq s \leq s_0$, the total number of comparisons is $\sum_{s=2}^{s_0}(n_s \sum_{k=1}^{s} k)$. In the arrow verification, the number of comparisons is $d-c$ because the cells indicated by the arrows are compared with other cells, *i.e.*, $d-c$ equals to the number of such other cells. In the adjacency verification, the protocol uses a coin placed on each of two cells separated by a bold line, shuffles the two coins, and compares them. Thus, the numbers of coins, shuffles, and comparisons are $2e$, e, and e, respectively. Finally, in the setup, the protocol compares coins placed on the same cell each other if there are multiple coins placed on the same cell. Since the room verification needs a coin placed on every cell, the number of comparisons is equal to the number of coins placed on cells for the other verifications. The adjacency verification needs such $2e$ coins placed on cells, and hence, the setup should perform $2e$ comparisons.

5.2 Security

We prove the security of the Makaro protocol. A proof for completeness is omitted because it is clear from the protocol description.

Lemma 7 (Perfect Completeness – Makaro). *If P knows a solution of a given Makaro grid, he can always convince V.*

Lemma 8 (Perfect Soundness – Makaro). *If P does not provide a collect solution of a given Makaro grid, V always rejects P.*

Proof. When P gives an incorrect solution, the following three situations are possible.

- A room of size s contains twice the same number, or a number not in $\{1, \ldots, s\}$. In this case, V will reject P's solution as in Sudoku (see Lemma 2).
- The number in the cell pointed by the arrow is not the highest. In this case, a comparison using the balance reveals that the coin in that cell is not the heaviest, and V rejects P's solution.
- Adjacent cells contains the same number. In this case, the balance will result in even when comparing the coins in these cells, and V will reject P's solution.

Therefore, when P does not give the correct answer, V will always reject. □

Lemma 9 (Perfect Zero-Knowledge – Makaro). *The verifier V is not given any information other than that the prover P can solve the Makaro grid.*

Proof. We show that no information has been leaked other than that the prover P can solve the Makaro grid through the following three checks in the verification phase:

- In step 1, when comparing the coins around the arrow cell, V does not learn anything except for which is the heaviest because the coins are visually indistinguishable and V never touches them and only observes the balance results.
- In step 2, V does not learn anything except that each room of size s contains one and only one number i for all $i \in \{1, \ldots, s\}$ for the exact same reason as for Sudoku (see Lemma 3).
- In step 3, when V checks that no adjacent cells contain the same number, as the coins are shuffled before the comparison and never reused after, V cannot learn anything except whether they are of different weight.

Therefore, V cannot learn anything throughout the whole process, except whether the solution is valid or not.

Formally, a simulator acts as follows.

- In the setup phase, the simulator places coins such that two identical coins are placed on the same cell if two coins should be placed on the same cell.
- In step 1, before comparing two coins, it swaps the coin placed on the cell pointed by an arrow with a coin of weight 2 and swaps the other coin with a coin of weight 1.
- In step 2(b), when s coins are shuffled, it swaps them with s coins weighted from 1 to s.
- In step 3, when two coins are shuffled, it swaps them with two coins of different weights.

This should make the comparison results performed in the protocol indistinguishable from those provided by an honest prover. □

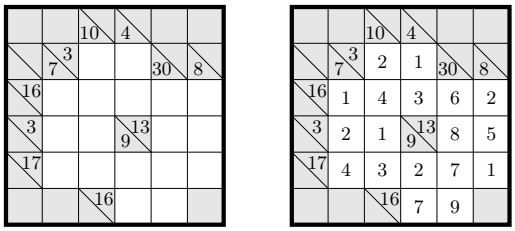

Fig. 6: An example of a Kakuro puzzle and its solution introduced in the Nikoli's website: https://www.nikoli.co.jp/en/puzzles/kakuro/.

6 Kakuro

Kakuro (or *Kakkuro*) was the most popular logic puzzle in Japanese printed press until 1992, when Sudoku took the top spot. The Kakuro grid has white cells and gray cells separated by diagonal lines into two triangular rooms. In Fig. 6, we give an example of a 6×6 Kakuro grid and its solution. The goal is to place one number in every white cells on the grid according to the following constraints:

1. The number in the upper right corner of the oblique line represents the sum of the numbers entering the consecutive white cells to its right.
2. The number in the lower left corner of the oblique line represents the sum of the numbers entering the consecutive white cells below it.
3. Each connected (*i.e.,* uninterrupted by a gray cell) row or column cannot contain twice the same number.

6.1 ZKP Protocol

By using a balance, it is easy to compare the numbers in the cells separated by diagonal lines with the sum of the numbers in the continuously connected cells. We detail our ZKP protocol. It achieves perfect completeness, perfect soundness and is perfectly zero-knowledge. The proofs are given in Sect. 6.2.

Setup: P and V fill the grid in two steps:

1. For each triangular cell, V places a coin ◯ of the indicated weight.
2. According to its solution, P places two coins ◯s on each white cell.

Verification: P and V execute the following steps:

1. For each triangular cell with a number, to verify if the weight of the coin on the cell is equal to the sum of the weights of the coins on consecutive white cells from the triangle cell, P and V follow these steps:
   - P compares the ◯ representing the number on the triangular cell with the ◯s on the consecutive white cells: ◯◯◯ | ◯.
   - If the comparison does not result in even, then V rejects P's solution.

- P moves the ◯s to their original positions.[6]

2. For each uninterrupted row (or column), V verifies that the coin placed on each cell is of a different weight than those placed on the other cells in the same way as steps 1 and 2 for Sudoku (Sect. 3).

Efficiency: Let t denote the number of triangular cells, n_ℓ the number of uninterrupted rows and columns of length ℓ, $2 \leq \ell \leq \ell_0$ for some ℓ_0, and w the number of white cells. This protocol uses a total of $t + 2w$ coins and performs t shuffles because in step 2, it applies a shuffle for each of uninterrupted rows and columns. The number of comparisons is $t + \sum_{\ell=2}^{\ell_0} \left(n_\ell \binom{\ell}{2} \right) + w$ because in step 2, it performs comparisons for all possible pairs of cells within each of uninterrupted rows and columns (and in step 1, a comparison is needed for each of triangular cells).

6.2 Security

We prove the security of the Kakuro protocol. A proof for completeness is omitted because it is clear from the protocol description.

Lemma 10 (Perfect Completeness – Kakuro). *If P knows a solution of a given Kakuro grid, he can always convince V.*

Lemma 11 (Perfect Soundness – Kakuro). *If P does not provide a correct solution of a given Kakuro grid, V always rejects P's solution.*

Proof. When P gives an incorrect solution, the following two situations are possible.

- A number in a triangular cell and the sum of the subsequent numbers from that triangle cell are not equal. In this case, the weight of the coin ◯ representing the triangular cell number and the sum of the weights of the coins ◯ in the consecutive white cells from that triangle cell are not equal, causing the balance to be unbalanced. Hence, V will reject P's solution.
- The same number is included twice in a block formed by consecutive white cells either vertically or horizontally. In this case, during the comparison of the coins ◯ in the white cells of the block, the weights of two coins are equal, resulting in the balance being even. Hence, V will reject P's solution.

Therefore, when P does not give the correct answer, V will always reject its solution. □

Lemma 12 (Perfect Zero-Knowledge – Kakuro). *The verifier V cannot learn any information other than that the prover P can solve the Kakuro grid.*

Proof. We show that no information has been leaked other than that the prover P can solve the Kakuro grid through each step of the verification phase:

[6] For this, P and V should memorize the order of ◯s when they are placed on the balance.

- In step 1, V checks that the number in the triangular cell is equal to the sum of the subsequent numbers. The coins ◯s from consecutive white cells are stacked before placing them on the scale. Hence V does not learn anything on their individual weight (they are visually indistinguishable), except that the sum of their weight is the same as the weight of the corresponding coin ◯.
- In step 2, V ensures that uninterrupted rows and columns do not contain twice the same number. The coins ◯s are shuffled, and each pair of coins is compared. Since the initial positions of each coin cannot be identified, V cannot determine the numbers on the white cells.

Hence, V cannot learn anything on P's solution throughout the whole protocol.

Formally, a simulator acts as follows.

- In the setup phase, the simulator arbitrarily places two identical coins on each white cell.
- In step 1 of the verification phase, before comparing, it swaps the coins placed on the consecutive white cells with the same number of coins such that the total weight of them is equal to the number on the triangular cell.
- In step 2, it acts in the same way as in the proof for Lemma 2.

This should make the comparison results performed in the protocol indistinguishable from those provided by an honest prover. □

7 Concluding Remarks

In this paper, we constructed ZKP protocols using a balance scale for four pencil-and-paper puzzles. We demonstrated the security of our proposed solutions, showing that they are perfectly complete, sound, and zero-knowledge. As a future work, we aim to explore other similar games. Additionally, we would like to investigate improvements that allow for the execution of the protocol with fewer coins and steps for the puzzles presented in this paper.

An analogous verification was considered in [7], where one confirms whether two cups contain the same number of marbles, say $X = Y$ or not. Because our model employs a balance to confirm which is heavier, say $X \geq Y$ or not, we considered an entirely different mechanism to construct a ZKP protocol. As can be observed from our ZKP protocols, Sudoku ZKP can be conducted only based on verifying $X = Y$ because it involves repeating the verification of whether a coin in a set is equal to each one in the other set.

Acknowledgements. 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 Number JP23H00479, the ANR project MobiS5 (ANR-18-CE39-0019), the ANR project SEVERITAS (ANR-20-CE39-0009) and the ANR Project PRIVA-SIQ (ANR-23-CE39-0008).

References

1. Abe, Y., Iwamoto, M., Ohta, K.: Efficient private PEZ protocols for symmetric functions. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. LNCS, vol. 11891, pp. 372–392. Springer, Cham (2019)
2. Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. Theor. Comput. Sci. **306**(1), 69–84 (2003)
3. Ben-Or, M., Goldreich, O., Goldwasser, S., Håstad, J., Kilian, J., Micali, S., Rogaway, P.: Everything provable is provable in zero-knowledge. In: CRYPTO 1988. LNCS, vol. 403, pp. 37–56. Springer, New York (1990)
4. Bultel, X., Dreier, J., Dumas, J., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: Demaine, E.D., Grandoni, F. (eds.) Fun with Algorithms. LIPIcs, vol. 49, pp. 8:1–8:20. Schloss Dagstuhl (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. Dreier, J., Jonker, H., Lafourcade, P.: Secure auctions without cryptography. In: Ferro, A., Luccio, F., Widmayer, P. (eds.) Fun with Algorithms. LNCS, vol. 8496, pp. 158–170. Springer, Cham (2014)
7. Glaser, A., Barak, B., Goldston, R.J.: A zero-knowledge protocol for nuclear warhead verification. Nature **510**, 497–502 (2014)
8. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. Journal of Cryptology **9**(3), 167–189 (1991)
9. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: Sedgewick, R. (ed.) STOC 1985. pp. 291–304. ACM (1985)
10. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. Theory of Computing Systems **44**(2), 245–268 (2009)
11. Hand, S., Koch, A., Lafourcade, P., Miyahara, D., Robert, L.: Check alternating patterns: A physical zero-knowledge proof for Moon-or-Sun. In: Shikata, J., Kuzuno, H. (eds.) IWSEC 2023. LNCS, vol. 14128, pp. 255–272. Springer (2023)
12. Hatsugai, K., Asano, K., Abe, Y.: A physical zero-knowledge proof for Sumplete, a puzzle generated by ChatGPT. In: Wu, W., Tong, G. (eds.) Computing and Combinatorics. LNCS, vol. 14422, pp. 398–410. Springer, Cham (2024)
13. Hearn, R.A., Demaine, E.D.: Games, Puzzles, and Computation. CRC Press (2009)
14. Iwamoto, C., Haruishi, M., Ibusuki, T.: Herugolf and Makaro are NP-complete. In: Ito, H., Leonardi, S., Pagli, L., Prencipe, G. (eds.) Fun with Algorithms. LIPIcs, vol. 100, pp. 24:1–24:11. Schloss Dagstuhl, Dagstuhl (2018)
15. Komano, Y., Mizuki, T.: Coin-based secure computations. Int. J. Inf. Secur. **21**, 833–846 (2022)
16. Lloyd, H., Crossley, M., Sinclair, M., Amos, M.: J-pop: Japanese puzzles as optimization problems. IEEE Transactions on Games **14**(3), 391–402 (2021)
17. Miyahara, D., Komano, Y., Mizuki, T., Sone, H.: Cooking cryptographers: Secure multiparty computation based on balls and bags. In: IEEE Computer Security Foundations Symposium. pp. 1–16. IEEE, NY (2021)
18. Miyahara, D., Robert, L., Lafourcade, P., Takeshige, S., Mizuki, T., Shinagawa, K., Nagao, A., Sone, H.: Card-based ZKP protocols for Takuzu and Juosan. In: Farach-Colton, M., Prencipe, G., Uehara, R. (eds.) Fun with Algorithms. LIPIcs, vol. 157, pp. 20:1–20:21. Schloss Dagstuhl (2021)

19. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for Kakuro. IEICE Trans. Fundamentals **102**(9), 1072–1078 (2019)
20. Mizuki, T., Kugimoto, Y., Sone, H.: Secure multiparty computations using a dial lock. In: Cai, J., Cooper, S.B., Zhu, H. (eds.) Theory and Applications of Models of Computation. LNCS, vol. 4484, pp. 499–510. Springer (2007)
21. Moran, T., Naor, M.: Basing cryptographic protocols on tamper-evident seals. Theor. Comput. Sci. **411**(10), 1283–1310 (2010)
22. Quisquater, J.J., Quisquater, M., Quisquater, M., Quisquater, M., Guillou, L., Guillou, M.A., Guillou, G., Guillou, A., Guillou, G., Guillou, S.: How to explain zero-knowledge protocols to your children. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 628–631. Springer, New York (1990)
23. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Card-based ZKP for connectivity: Applications to Nurikabe, Hitori, and Heyawake. New Gener. Comput. **40**(1), 149–171 (2022)
24. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Hide a liar: Card-based ZKP protocol for Usowan. In: Du, D., Du, D., Wu, C., Xu, D. (eds.) Theory and Applications of Models of Computation. LNCS, vol. 13571, pp. 201–217. Springer (2022)
25. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Physical ZKP protocols for Nurimisaki and Kurodoko. Theor. Comput. Sci. **972**, 114071 (2023)
26. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for Sudoku. New Gener. Comput. **40**(1), 49–65 (2022)
27. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Ripple Effect. Theor. Comput. Sci. **895**, 115–123 (2021)
28. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for Sudoku. Theor. Comput. Sci. **839**, 135–142 (2020)
29. Shinagawa, K., Mizuki, T., Schuldt, J.C.N., Nuida, K., Kanayama, N., Nishide, T., Hanaoka, G., Okamoto, E.: Secure computation protocols using polarizing cards. IEICE Trans. Fundamentals **99-A**, 1122–1131 (2016)
30. Tanaka, K., Mizuki, T.: Two uno decks efficiently perform zero-knowledge proof for Sudoku. In: Fernau, H., Jansen, K. (eds.) Fundamentals of Computation Theory. LNCS, vol. 14292, pp. 406–420. Springer, Cham (2023)
31. Uehara, R.: Computational complexity of puzzles and related topics. Interdisciplinary Information Sciences **29**(2), 119–140 (2023)
32. Yato, T., Seta, T.: Complexity and completeness of finding another solution and its application to puzzles. IEICE Trans. Fundamentals **86**(5), 1052–1060 (2003)