Two UNO Decks Efficiently Perform Zero-Knowledge Proof for Sudoku*

Kodai Tanaka[1] and Takaaki Mizuki[2]

[1] Graduate School of Information Sciences, Tohoku University, Sendai, Japan
[2] Cyberscience Center, Tohoku University, Sendai, Japan

Abstract. Assume that there is a challenging Sudoku puzzle such that a prover knows a solution while a verifier does not know any solution. A zero-knowledge proof protocol allows the prover to convince the verifier that the prover knows the solution without revealing any information about it. In 2007, Gradwohl et al. constructed the first physical zero-knowledge proof protocol for Sudoku using a physical deck of playing cards; its drawback would be to have a soundness error. In 2018, Sasaki et al. improved upon the previous protocol by developing soundness-error-free protocols; their possible drawback would be to require many standard decks of playing cards, namely nine (or more) decks. In 2021, Ruangwises designed a novel protocol using only two standard decks of playing cards although it requires 322 shuffles, making it difficult to use in practical applications. In this paper, to reduce both the numbers of required decks and shuffles, we consider the use of UNO decks, which are commercially available: we propose a zero-knowledge proof protocol for Sudoku that requires only two UNO decks and 16 shuffles. Thus, the proposed protocol uses reasonable numbers of decks and shuffles, and we believe that it is efficient enough for humans to execute practically.

Keywords: Card-based cryptography, Sudoku, Zero-knowledge proof

1 Introduction

Sudoku is one of the most popular pencil puzzles in the world. A standard Sudoku puzzle consists of a 9×9 grid divided into 9 sub-grids (whose sizes are 3×3); we call such a sub-grid a *block* throughout this paper. Some of the cells on the grid are already filled with numbers from 1 through 9, and the goal of the puzzle is

* Sudoku and UNO are trademarks or registered trademarks of Nikoli Co., Ltd. and Mattel, Inc., respectively.

This paper appears in Proceedings of FCT 2023. This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-3-031-43587-4_29. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use: https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms.

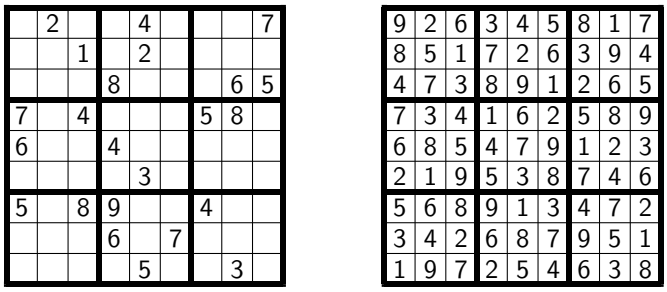


Fig. 1: Example of a standard Sudoku puzzle and its solution

to place a number on each empty cell so that exactly one number from 1 to 9 appears in each row, each column, and each block. Figure 1 shows an example of a puzzle instance and its solution.

1.1 Zero-knowledge Proof for Sudoku

Assume that a player P had created a challenging Sudoku puzzle and P showed the puzzle to another player V. The player V has spent much time and effort finding a solution, but V cannot find any solution. The player V gets skeptical about whether the puzzle really has a solution. As easily imaged, this is the right opportunity to make use of the *zero-knowledge proof* [3]. Thus, this paper deals with zero-knowledge proof protocols for Sudoku.

In addition, we solicit *physical* zero-knowledge proof protocols that do not use any electronic devices such as computers, but use everyday objects such as a physical deck of playing cards. Physical protocols tend to be simple and easy-to-understand so that lay-people can easily execute them.

1.2 The Existing Protocols

In 2007, for the first time, Gradwohl et al. [4,5] developed several physical zero-knowledge proof protocols for Sudoku. Among them, the most easy-to-implement protocol would be the one which uses a physical deck of playing cards. This protocol is the first *card-based* zero-knowledge proof protocol for Sudoku; roughly speaking, after a prover P places face-down cards to commit to the solution P has, the protocol applies a series of actions, such as shuffling and turning over cards, to convince a verifier V that P surely knows the solution. Since the protocol has a soundness error, it must be repeated many times to make the soundness error probability negligibly small.

Later in 2018, Sasaki et al. [28] improved upon the previous protocol by devising a new technique (called the "uniqueness verification protocol" as will be seen in Sect. 2.3) to eliminate the soundness error. Specifically, they proposed three zero-knowledge proof protocols for Sudoku having no soundness error [27]; we name them *Sasaki's Protocols A*, *B*, and *C*, whose performances are shown

Table 1: Comparison of the proposed protocol with the existing protocols

Protocols	Number of cards	Accommodating decks	Number of shuffles	Interactive?
Sasaki's Protocol A [27]	90	9 standard decks 5 UNO decks	45	No
Sasaki's Protocol B [27]	171	18 standard decks 9 UNO decks	36	No
Sasaki's Protocol C [27]	243	27 standard decks 14 UNO decks	28	Yes
Ruangwises's Protocol A [22]	120	3 standard decks	108	Yes
Ruangwises's Protocol B [22]	108	2 standard decks	322	Yes
Ours	117	2 UNO decks	16	No

as the first three protocols listed in Table 1. For instance, Sasaki's Protocol A uses 9 sets of 9 numbered cards $\boxed{1}\boxed{2}\boxed{3}\boxed{4}\boxed{5}\boxed{6}\boxed{7}\boxed{8}\boxed{9}$ as well as arbitrary distinct 9 cards, and hence, it requires 90 cards in total and all the required cards can be accommodated by 9 standard decks of playing cards. As for the number of shuffles, it applies a "pile-scramble shuffle" (which is a kind of a shuffling action as seen later in Sect. 2.2) 45 times. As known from Table 1, Sasaki's protocols require at least 9 standard decks and at least 28 shuffles. Note that Sasaki's Protocols A and B are "non-interactive" whereas Sasaki's Protocol C is "interactive."[iii] A *non-interactive* card-based zero-knowledge proof protocol uses no prover's knowledge after a prover commits to a solution with face-down cards (and hence, the protocol can be executed by either a prover or a verifier, or even a third party) [12]. By contrary, an *interactive* card-based zero-knowledge proof protocol requires prover's knowledge during the execution of the protocol.

In 2021, to reduce the number of required standard decks, Ruangwises [21,22] proposed two novel interactive protocols, which we name *Ruangwises's Protocols A* and *B*, using two or three standard decks of playing cards. For instance, Ruangwises's Protocol B requires two sets of $13 \times 4 = 52$ cards

$$\boxed{\clubsuit A}\boxed{\diamondsuit A}\boxed{\heartsuit A}\boxed{\spadesuit A}\boxed{\clubsuit 2}\boxed{\diamondsuit 2}\boxed{\heartsuit 2}\boxed{\spadesuit 2}\cdots\boxed{\clubsuit K}\boxed{\diamondsuit K}\boxed{\heartsuit K}\boxed{\spadesuit K}$$

along with two jokers (i.e., two cards of different patterns); therefore, two standard decks can accommodate all the required cards. Thus, Ruangwises's Protocol B is very efficient in terms of the number of required decks. However, the number of required shuffles is 322, which is too many for humans to execute. Overall, as known from Table 1, Ruangwises's protocols use only two or three decks while requiring at least 108 shuffles.

1.3 Contribution of This Paper

In this paper, instead of standard decks, we consider the use of UNO decks, which are also commercially available over the world. Specifically, we construct

[iii] The usage of these terms is valid only for card-based zero-knowledge proof protocols.

(a) Yellow, red, and blue numbered cards

(b) 27 distinct cards

Fig. 2: UNO cards to use

a zero-knowledge proof protocol for Sudoku that works on two UNO decks using only 16 shuffles. Our proposed protocol is non-interactive (and has no soundness error); see the last line in Table 1.

The specific numbers of required cards are as follows. We use four sets of yellow numbered cards

Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9,

three sets of red numbered cards

R1 R2 R3 R4 R5 R6 R7 R8 R9,

and three sets of blue numbered cards

B1 B2 B3 B4 B5 B6 B7 B8 B9

(as partially illustrated in Fig. 2(a)) along with 27 arbitrary distinct cards (as illustrated in Fig. 2(b)). One can easily confirm that two UNO decks can accommodate all the necessary cards mentioned above.

Making use of "color information" of cards, as will be seen, we can aggregate necessary shuffles, resulting in only the 16 shuffles. The shuffling operation our protocol uses is also the pile-scramble shuffle; our protocol uses such a shuffle for 27 cards whereas the existing ones use it for 9 cards or less. Note that a pile-scramble shuffle for 27 cards is also easy-to-implement.

Needless to say, zero-knowledge proofs play an important role in providing security and privacy. As Hanaoka [6] mentioned, for technology diffusion, we require easy-to-understand structures of cryptographic tools whereby potential users can easily understand the essential mechanisms. Since Sudoku is one of the most famous puzzles and our protocol is quite simple, our protocol might motivate potential users to try to use zero-knowledge proof technology.

1.4 Related Work

Aside from Sudoku, many physical zero-knowledge proof protocols have been constructed for other pencil puzzles. Examples are as follows: Akari [1], Hashiwokakero (Bridges) [25], Heyawake [19], Kakuro [1,14], Makaro [2,26], Masyu [10],

Numberlink [23, 24], Nurikabe [18, 19], Slitherlink [10, 11], Suguru [17, 20], and Takuzu [1, 13].

2 Preliminaries

In this section, we formally describe the cards and actions on them, and explain the "pile-scramble shuffle." Then, we introduce the "uniqueness verification protocol," which we will use as a sub-protocol in our protocol.

2.1 Cards and Actions

First, as mentioned in Sect. 1.3, we use numbered cards of three colors (yellow, red, and blue); the face of every card has a number from 1 through 9 and the patterns of the backs of all cards are identical, illustrated as

face up: R1 Y6 B9, face down: ? ? ?.

Specifically, as mentioned before, we use four sets of yellow numbered cards Y1 Y2 ⋯ Y9, three sets of red numbered cards R1 R2 ⋯ R9, and three sets of blue numbered cards B1 B2 ⋯ B9. We call these cards *encoding cards*, which we will use to represent a solution to a Sudoku puzzle.

In addition to the encoding cards, 27 distinct cards h_1 h_1 ⋯ h_{27} with the identical backs ? ? ⋯ ? are used; these are called *helping cards*. We choose helping cards from UNO decks as already illustrated in Fig. 2(b).

Our proposed protocol follows the standard computational model of card-based cryptographic protocols [8,9,15,16], in which a protocol is formally defined as an abstract machine. Here, we introduce three main actions, which are applied to a sequence of cards; below, we assume a sequence of m cards.

Permute.
This is denoted by (perm, π), where π is a permutation applied to the sequence of cards as follows:

$$\overset{1}{?}\ \overset{2}{?}\ \cdots\ \overset{m}{?} \xrightarrow{(\mathsf{perm},\pi)} \overset{\pi^{-1}(1)}{?}\ \overset{\pi^{-1}(2)}{?}\ \cdots\ \overset{\pi^{-1}(m)}{?}.$$

Turn.
This is denoted by (turn, T), where T is a set of indices, indicating that for every $t \in T$, the t-th card is turned over as follows:

$$\overset{1}{?}\ \overset{2}{?}\ \cdots\ \overset{t\in T}{?}\ \cdots\ \overset{m}{?} \xrightarrow{(\mathsf{turn},T)} \overset{1}{?}\ \overset{2}{?}\ \cdots\ \overset{t\in T}{\mathrm{R1}}\ \cdots\ \overset{m}{?}.$$

Shuffle.
This is denoted by (shuf, Π), where Π is a set of permutations, indicating that $\pi \in \Pi$ is drawn uniformly and is applied to the sequence of cards as follows:

$$\overset{1}{?}\ \overset{2}{?}\ \cdots\ \overset{m}{?} \xrightarrow{(\mathsf{shuf},\Pi)} \overset{\pi^{-1}(1)}{?}\ \overset{\pi^{-1}(2)}{?}\ \cdots\ \overset{\pi^{-1}(m)}{?}.$$

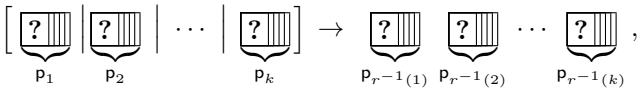
Although a protocol is supposed to be defined with a combination of the actions above (or an abstract machine), in the sequel, we use a natural language to describe a protocol for simplicity.

2.2 Pile-Scramble Shuffle

In our protocol, we use the *pile-scramble shuffle*, which was devised in [7].

Assume that there are k piles of cards, denoted by p_i for every i, $1 \leq i \leq k$, such that all the piles have the same number of cards:

A pile-scramble shuffle is a shuffling action that applies a random permutation $r \in S_k$ to the sequence of piles, denoted by $[\cdot|\cdots|\cdot]$:



where S_k denotes the symmetric group of degree k.

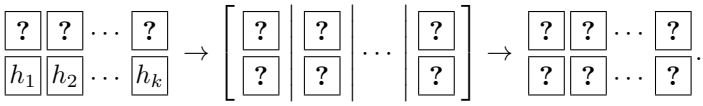
A pile-scramble shuffle can be implemented in practice by placing each pile of cards in a sleeve or envelope and stirring the envelopes (until the original sequence is no longer discernible).

2.3 Uniqueness Verification Protocol

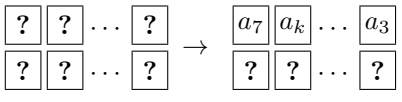
This subsection explains the *uniqueness verification protocol* using the pile-scramble shuffle described in Sect. 2.2. This protocol has been proposed by Sasaki et al. [27,28].

Assume that there is a sequence of k face-down cards $\boxed{?}\boxed{?}\cdots\boxed{?}$ such that it consists of k distinct cards $\boxed{a_1}\boxed{a_2}\cdots\boxed{a_k}$ arranged in an unknown order. The uniqueness verification protocol enables us to confirm that the sequence surely consists of these k distinct cards without revealing the unknown order while keeping the original sequence unchanged. The protocol uses k helping cards $\boxed{h_1}\boxed{h_2}\cdots\boxed{h_k}$ and proceeds as follows.

1. Place the k helping cards below the unknown sequence, turn them face down, and apply a pile-scramble shuffle as follows:



2. Turn over all the cards of the top sequence to verify that it consists of $\{a_1, a_2, \ldots, a_k\}$:



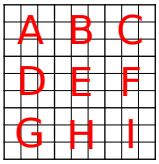


Fig. 3: The names of the 9 blocks

3. After turning over all the cards of the top sequence, apply a pile-scramble shuffle:

$$\left[\begin{array}{c|c|c|c} ? & ? & \cdots & ? \\ ? & ? & & ? \end{array}\right] \rightarrow \begin{array}{cccc} ? & ? & \cdots & ? \\ ? & ? & \cdots & ? \end{array}.$$

4. Turn over all the cards of the bottom sequence and sort the piles so that the bottom sequence becomes $h_1, h_2, \ldots, h_k$ in this order:

$$\begin{array}{cccc} ? & ? & \cdots & ? \\ h_2 & h_5 & \cdots & h_3 \end{array} \rightarrow \begin{array}{cccc} ? & ? & \cdots & ? \\ h_1 & h_2 & \cdots & h_k \end{array}.$$

This restores the original sequence.

As mentioned above, we utilize this uniqueness verification protocol as a sub-protocol when constructing our protocol.

3 Building Blocks

Our proposed protocol consists of several sub-protocols: the color verification, row verification, color change, and column verification sub-protocols. In this section, we present these sub-protocols as building blocks for our proposed protocol (which will be shown in the next section). Hereinafter, we call the 9 blocks of a Sudoku puzzle *Blocks* $A, B, C, \ldots, H, I$ as shown in Fig. 3.

We first explain how a prover P commits to a solution with face-down encoding cards in Sect. 3.1, and then the sub-protocols will be presented in the succeeding subsections.

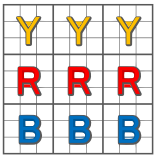
3.1 Commitment to a Solution

Assume that a prover knows a solution of a given 9×9 Sudoku puzzle. According to the solution, the prover P secretly arranges face-down encoding cards as follows.

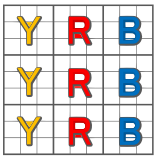
1. Prepare three sets of encoding cards Y1 Y2 $\cdots$ Y9 R1 R2 $\cdots$ R9 B1 B2 $\cdots$ B9.

(a) An actual placement by P

(b) The placement after turning over face-up cards



(c) Color scheme for input



(d) After applying the color change protocol

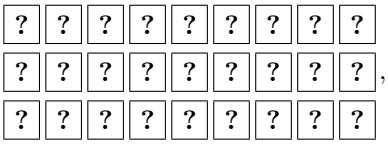
Fig. 4: Commitment to a solution by a prover and color schemes for input blocks and those after color change

2. P and V (or someone) publicly place an encoding card on each cell that already has a number written on it such that the number of the card is equal to the written number (on the cell) and its color matches the color scheme for blocks shown in Fig. 4(c) (i.e., yellow cards are placed in Blocks A, B, and C, red cards in Blocks D, E, and F, and blue cards in Blocks G, H, and I).
3. According to the solution, P secretly places face-down encoding cards on the remaining empty cells (without V's seeing the faces), based on the same color and number schemes as in the previous step.

In this way, the prover P commits to the solution with face-down encoding cards; Fig. 4(a) illustrates an actual placement by P, and Fig. 4(b) shows the placement after turning over all the face-up cards, which we sometimes call a *commitment* to the solution.

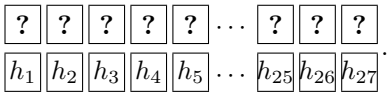
3.2 Color Verification Sub-protocol

Given a commitment to a solution (as illustrated in Fig. 4(b)), we first want to check that the face-down cards satisfy the color scheme properly. That is, given three rows of length 9:



we want to verify that all cards have the same color, say yellow. More precisely, the following *color verification sub-protocol* can confirm that the 27 cards consist of three sets of Y1 Y2 $\cdots$ Y9 using 27 helping cards.

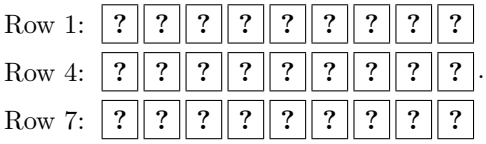
1. Make a sequence of length 27 from the three rows, and place the 27 helping cards below it:



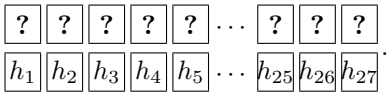
2. Apply the uniqueness verification protocol described in Sect. 2.3. If the opened cards do not consist of three sets of Y1 Y2 ⋯ Y9, the protocol aborts.
3. Move the face-down cards (of the top sequence) back to the original positions to restore the three rows.

3.3 3-row Verification Sub-protocol

Suppose that all the cards in Blocks A, B, and C are yellow, all the cards in Blocks D, E, and F are red, and all the cards in Blocks G, H, and I are blue. The 3-*row verification sub-protocol* can verify that each of the three rows (of different colors) consists of numbers 1 through 9. Take Rows $1, 4$, and 7 as an example for row verification:



1. Make a sequence of 27 cards from the three row and place 27 helping cards below it:



2. Apply the uniqueness verification protocol described in Sect. 2.3. If the opened cards do not consist of

the protocol aborts.
3. Move the face-down cards (of the top sequence) back to the original positions to restore the three rows.

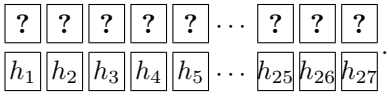
The same can be applied to the other three rows (of different colors).

3.4 Color Change Sub-protocol

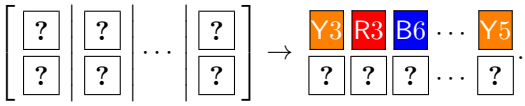
In this subsection, we present the *color change sub-protocol*, which changes only the colors of the cards in blocks without changing their numbers as well as provides block verification.

Assume that there are three blocks, the first one consisting of yellow cards, the second one consisting of red cards, and the third one consisting of blue cards. The following procedure exchanges the colors of the first and second blocks as well as changes the color of the third block into yellow[iv].

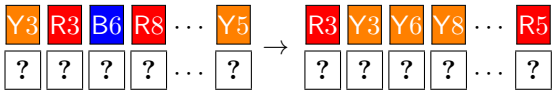
1. Make a sequence of 27 cards from the three blocks and place 27 helping cards below it:



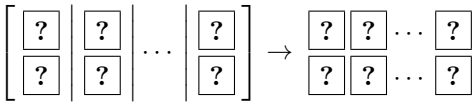
2. Apply a pile-scramble shuffle to them. Turn the encoding cards face up, making sure that the yellow, red, and blue cards from 1 to 9 appear without omission (otherwise it aborts):



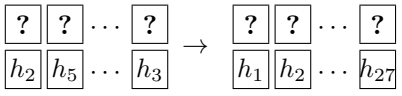
3. For yellow and red cards, swap the cards with the same number (for example, swap the positions of the yellow 3 and the red 3). For blue cards, preparing 9 free yellow cards Y1 Y2 ⋯ Y9,[v] swap the cards with the same number:



4. Turn the encoding cards face down and apply a pile-scramble shuffle:



5. Turn over all the cards of the bottom sequence and sort the piles so that the bottom sequence become $h_1, h_2, \ldots, h_{27}$ in this order. Return the encoding cards to their original positions:

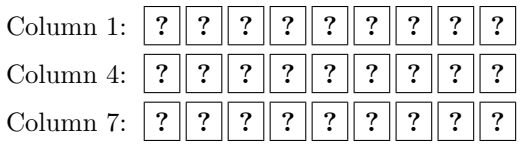


Note that this sub-protocol also checks that each of the three blocks consists of numbers from 1 to 9.

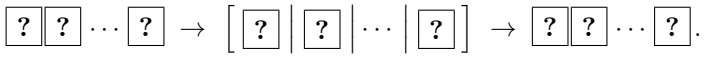
[iv] The sub-protocol works for other combinations of colors.
[v] This is why our protocols needs four sets of yellow cards.

3.5 3-column Verification Sub-Protocol

Suppose that all the cards in Blocks A, D, and G are yellow, all the cards in Blocks B, E, and H are red, and all the cards in Blocks C, F, and I are blue. The 3-*column verification sub-protocol* can verify that each of the three columns (of different colors) consists of numbers 1 through 9 while it does not revert the blocks contrary to the 3-row verification sub-protocol presented in Sect. 3.3. Take Columns 1, 4, and 7 as an example for column verification:



1. Make a sequence of 27 cards from the three columns and apply a pile-scramble shuffle (i.e., a normal shuffle):



2. Turn over all the 27 cards. If the opened cards do not consist of

the protocol aborts.

Since this sub-protocol will be used at the last step of our proposed protocol, there is no need to restore the three retrieved columns. Therefore, the uniqueness verification protocol described in Sect. 2.3 is not used. The same can be applied to other three columns (of different colors).

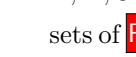
4 Our Protocol for 9×9 Sudoku

In this section, we construct a zero-knowledge proof protocol for a standard Sudoku puzzle.

A prover P who knows a solution places a commitment to the solution as described in Sect. 3.1, and this is the input to the protocol. After the input is given, the protocol does not require the knowledge of the prover P and can be executed by anyone thereafter (because our protocol is non-interactive).

Given the placement by P along with 27 helping cards and free 9 cards 

, our protocol proceeds as follows.

1. Apply the color verification sub-protocol given in Sect. 3.2 to each of Blocks A, B, C, Blocks D, E, F, and Blocks G, H, I. If it does not abort, Blocks A, B, C consist of three sets of 

, Blocks D, E, F consist of three sets of

, and Blocks G, H, I consist of three sets of

.

2. Apply the 3-row verification sub-protocol given in Sect. 3.3 to Rows $1, 4$, and 7. Apply also the 3-row verification sub-protocol to Rows $2, 5$, and 8. If it does not abort, Rows $1, 2, 4, 5, 7$, and 8 consist of numbers 1 through 9. In addition, this result together with the result of the color verification in the previous step automatically implies that the remaining Rows $3, 6$, and 9 must consist of numbers 1 through 9.
3. In order to be able to verify the columns, we want to change the color of Blocks A, D, and G to yellow, Blocks B, E, and H to red, and Blocks C, F, and I to blue, as shown in Fig. 4(d). First, apply the color change sub-protocol described in Sect. 3.4 to Blocks B, D, and G with colors yellow, red, and blue in this order. This swaps the colors of Blocks B and D, and changes the color of Block G to yellow. (This produces 9 free cards $\boxed{\text{B1}}\,\boxed{\text{B2}}\,\cdots\,\boxed{\text{B9}}$.) Second, apply the color change sub-protocol to Blocks H, F, and C with colors blue, red, and yellow in this order. This swaps the colors of Blocks H and F, and changes the color of Block C to blue. Through these color transformations, Blocks B, C, D, F, G, and H are verified to consist of numbers 1 through 9. In addition, based on the results of Step 1, it is automatically guaranteed that the remaining Blocks A, E, and I consist of numbers 1 through 9.
4. Apply the 3-column verification sub-protocol described in Sect. 3.5 to Columns $1, 4$, and 7. The same procedure is applied to Columns $2, 5$, and 8. The remaining Columns $3, 6$, and 9 are automatically verified as in the row case.

This is the proposed protocol for convincing a verifier that the numbers 1 to 9 appear exactly once in each row, each column, and each block.

The number of required cards is 117 and the number of shuffles is 16 although we omit the breakdowns due to the space limitation. We also omit the security proof in this extended abstract.

5 Conclusion

In this paper, we proposed a new card-based zero-knowledge proof protocol for Sudoku. The main idea behind our proposed protocol is to make use of UNO decks, which are commercially available over the world, and hence, they are easy to prepare. The property that there are several UNO cards having the same pair of a color and a number on their faces leads to reducing the number of required shuffles. Specifically, two UNO decks and 16 shuffles are sufficient for constructing a zero-knowledge proof protocol for a standard Sudoku puzzle. We believe that this is efficient enough for humans to execute practically. The proposed protocol can also be implemented with standard decks of playing cards although it requires four decks along with two jokers of different designs.

Our techniques can be applied to a general $n \times n$ Sudoku although we omit the details in this extended abstract.

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 Numbers JP21K11881 and JP23H00479.

References

1. Bultel, X., Dreier, J., Dumas, J.G., Lafourcade, P.: Physical zero-knowledge proofs for Akari, Takuzu, Kakuro and KenKen. In: Fun with Algorithms. LIPIcs, vol. 49, pp. 8:1–8:20. Schloss Dagstuhl, Dagstuhl, Germany (2016)
2. 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: Stabilization, Safety, and Security of Distributed Systems. LNCS, vol. 11201, pp. 111–125 (2018)
3. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Annual ACM Symposium on Theory of Computing. pp. 291–304. STOC'85, ACM, New York (1985)
4. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. In: Fun with Algorithms. LNCS, vol. 4475, pp. 166–182. Springer, Berlin, Heidelberg (2007)
5. 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)
6. Hanaoka, G.: Towards user-friendly cryptography. In: Paradigms in Cryptology–Mycrypt 2016. Malicious and Exploratory Cryptology. LNCS, vol. 10311, pp. 481–484. Springer, Cham (2017)
7. Ishikawa, R., Chida, E., Mizuki, T.: Efficient card-based protocols for generating a hidden random permutation without fixed points. In: Unconventional Computation and Natural Computation. LNCS, vol. 9252, pp. 215–226. Springer, Cham (2015)
8. 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: Advances in Cryptology—ASIACRYPT 2017. LNCS, vol. 10626, pp. 126–155. Springer, Cham (2017)
9. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Advances in Cryptology—ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015)
10. Lafourcade, P., Miyahara, D., Mizuki, T., Robert, L., Sasaki, T., Sone, H.: How to construct physical zero-knowledge proofs for puzzles with a "single loop" condition. Theor. Comput. Sci. **888**, 41–55 (2021)
11. Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: A physical ZKP for Slitherlink: How to perform physical topology-preserving computation. In: Information Security Practice and Experience. LNCS, vol. 11879, pp. 135–151. Springer, Cham (2019)
12. Miyahara, D., Haneda, H., Mizuki, T.: Card-based zero-knowledge proof protocols for graph problems and their computational model. In: Provable and Practical Security. LNCS, vol. 13059, pp. 136–152. Springer, Cham (2021)

13. 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: Fun with Algorithms. LIPIcs, vol. 157, pp. 20:1–20:21. Schloss Dagstuhl, Dagstuhl, Germany (2020)
14. Miyahara, D., Sasaki, T., Mizuki, T., Sone, H.: Card-based physical zero-knowledge proof for Kakuro. IEICE Trans. Fundam. **102**(9), 1072–1078 (2019)
15. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. Int. J. Inf. Secur. **13**(1), 15–23 (2014)
16. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Trans. Fundam. **E100.A**(1), 3–11 (2017)
17. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Physical zero-knowledge proof for Suguru puzzle. In: Stabilization, Safety, and Security of Distributed Systems. LNCS, vol. 12514, pp. 235–247. Springer, Cham (2020)
18. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Interactive physical ZKP for connectivity: Applications to Nurikabe and Hitori. In: Connecting with Computability. LNCS, vol. 12813, pp. 373–384. Springer, Cham (2021)
19. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Card-based ZKP for connectivity: Applications to Nurikabe, Hitori, and Heyawake. New Gener. Comput. **40**, 149–171 (2022)
20. Robert, L., Miyahara, D., Lafourcade, P., Libralesso, L., Mizuki, T.: Physical zero-knowledge proof and NP-completeness proof of Suguru puzzle. Inf. Comput. **285**, 1–14 (2022)
21. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for Sudoku. In: Computing and Combinatorics. LNCS, vol. 13025, pp. 631–642. Springer, Cham (2021)
22. Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for Sudoku. New Gener. Comput. **40**, 49–65 (2022)
23. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Numberlink. In: Fun with Algorithms. LIPIcs, vol. 157, pp. 22:1–22:11. Schloss Dagstuhl, Dagstuhl, Germany (2020)
24. Ruangwises, S., Itoh, T.: Physical zero-knowledge proof for Numberlink puzzle and k vertex-disjoint paths problem. New Gener. Comput. **39**(1), 3–17 (2021)
25. Ruangwises, S., Itoh, T.: Physical ZKP for connected spanning subgraph: Applications to Bridges puzzle and other problems. In: Unconventional Computation and Natural Computation. pp. 149–163. Springer, Cham (2021)
26. Ruangwises, S., Itoh, T.: Physical ZKP for Makaro using a standard deck of cards. In: Theory and Applications of Models of Computation. LNCS, vol. 13571, pp. 43–54. Springer, Cham (2022)
27. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for Sudoku. Theor. Comput. Sci. **839**, 135–142 (2020)
28. Sasaki, T., Mizuki, T., Sone, H.: Card-based zero-knowledge proof for Sudoku. In: Fun with Algorithms. LIPIcs, vol. 100, pp. 29:1–29:10. Schloss Dagstuhl, Dagstuhl, Germany (2018)