一般化数独に対する効率的な天秤ベースゼロ知識証明

金子 尚平[1,a)]　崎山 一男[1]　宮原 大輝[1,2]

概要：数独の解に対するゼロ知識証明は、ある与えられた数独の問題に対して、解に関する情報を一切漏らすことなく、その解が存在することを検証者に証明する手法である。これまでに数独を含む多くのパズルに対して、カード組を用いたゼロ知識証明プロトコルが提案されている。これらのカードベースの手法に対して、著者らは新たに天秤とコインを用いたゼロ知識証明方式を提案してきた。この方式では、証明者がコインの重さによって数独の解を表現し、各行/列/ブロックに 1 から 9 に対応する異なる重さのコインが存在することを、1 枚ずつ比較操作を行うことで解の存在だけを示す。本研究では、この既存の天秤ベースゼロ知識証明プロトコルを拡張し、盤面のサイズが $n \times n$ である一般化数独に対して構成を記述する。さらに、従来は各コインを個別に検証していたのに対し、本稿では行の検証が完了した後、列やブロックに対しては、それぞれ n 枚のコインをまとめて重ね、コインの合計重量のみを検証することで、必要な比較回数を大幅に削減する新たなプロトコルを提案する。これにより比較回数は従来の方式と比べて約 1/3 となり、効率性が大きく向上する。

キーワード：カードベース暗号、ゼロ知識証明、天秤、ペンシルパズル、数独

Efficient Balance-Based ZKP Protocol for Generalized Sudoku

Shohei Kaneko[1,a)]　Kazuo Sakiyama[1]　Daiki Miyahara[1,2]

Abstract: A zero-knowledge proof for a Sudoku solution allows a prover to convince a verifier of the existence of a valid solution to a given Sudoku puzzle, without revealing any information about the solution itself. In previous research, a number of zero-knowledge proof protocols using a deck of physical cards have been proposed for Sudoku and various other pencil-and-paper puzzles. In contrast to these card-based approaches, we have introduced a new zero-knowledge proof system using a balance scale and coins. In this approach, the prover represents a Sudoku solution using coins of different weights corresponding to numbers 1 through 9, and demonstrates that each row, column, and block contains coins of all distinct weights by performing comparisons. In this study, we extend the existing balance-based zero-knowledge protocol to generalized Sudoku puzzles of size $n \times n$ and present its pesudocode. Furthermore, unlike prior protocols that verify each coin individually, the proposed method performs individual verification only for rows, and then verifies each of columns and blocks by stacking the n corresponding coins and comparing their total weight. This allows the verifier to check whether the total weight matches the expected sum. As a result, the number of required comparisons is reduced to approximately one-third of that in prior protocols, significantly improving the efficiency.

Keywords: Card-based Cryptography, Zero-Knowldge Proof, Balance Scale, Pencil-and-Paper Puzzle, Sudoku

1. 始めに

ゼロ知識証明は、ある命題が真であることをそれ以外の情報を一切明かすことなく、検証者に納得させる方法である。数独の解に対するゼロ知識証明は、与えられた数独の問題に対して、解に関する情報を一切明かさずに、解が存在することを検証することを可能にする。数独は、世界的に広く認知されているパズルゲームであり、ルールが単純であることからゼロ知識証明の仕組みを具体的に示す題材として優れている。

[1] 電気通信大学
[2] 産業技術総合研究所
[a)] shohei.kaneko@uec.ac.jp

数独に対する物理的なゼロ知識証明として、2009 年に Gradwhol ら [3] は、トランプのような物理的なカード組を用いるプロトコルを提案した。その後、数独以外の多くのパズルにも拡張され、カード組を用いたゼロ知識証明プロトコルが多く提案されている。数独を対象とした研究では、必要なカード枚数やシャッフル回数を削減する効率化手法 [7,9–11] が提案されている。

1.1 天秤ベースゼロ知識証明

著者らは、従来のカードベースゼロ知識証明プロトコルとは異なる新たな物理的道具として、**天秤**を用いたゼロ知識証明プロトコル [4,5] を提案してきた。天秤ベースのゼロ知識証明では、外見から重さを識別できないコインと天秤を用い、その重さを比較することでパズルの解を検証する。これまでに著者らは、この方式を数独のほか、マカロ、カックロ、不等式といったパズル [4] に適用してきた。また、数独に対する天秤ベースのゼロ知識証明プロトコルについては、使用するコインの枚数や天秤による比較回数を削減し、効率性を向上させた改良プロトコル [5,6] も提案している。

1.2 一般化された数独

市販の雑誌に掲載されている数独は大抵、9×9 マスで構成され、さらにそれらのマスが 3×3 の 9 個のブロックに分割されている。数独の目的は、各行/列/ブロックに 1 から 9 までの数字が 1 回ずつ現れるように数字を埋めることである。これは n を完全平方数とする $n \times n$ マスの盤面に拡張でき、本稿ではこれを**一般化数独**と呼ぶ。一般化数独では、盤面は $\sqrt{n} \times \sqrt{n}$ の n 個のブロックに区切られ、各行/列/ブロックに 1 から n までの数字が 1 回ずつ現れるように数字を埋めることが目的となる。一般化数独の判定問題は NP 完全であることが証明されている [12]。

1.3 貢献

本稿では、数独に対する既存の天秤ベースゼロ知識証明 [6] を $n \times n$ に拡張された一般化数独に対して形式的に記述する。技術的な貢献としては、複数枚のコインを天秤に載せて検証する “天秤らしい” 方法を提案する。表 1 に、一般化数独に対する既存プロトコルと提案プロトコルを実行する際に必要なコインの枚数とシャッフル回数を示す。既存プロトコルでは、各行・列・ブロックに 1 から n に対応する異なる重さのコインが存在することを、1 枚ずつ比較して検証していた。これに対し本稿では、行の検証完了後、列およびブロックについては対応する n 枚のコインをまとめて重ね、その合計重量のみを比較する。これにより、列およびブロックの検証はいずれも 1 回の比較で済み、必要な比較回数は従来の約 1/3 に削減される。

提案プロトコルを含む物理的な道具を用いるゼロ知識証明の多くは Covert Security [1] を念頭に置き、証明者は検証者に露見しない範囲であらゆる不正を働くとしても、健全性が満たされるようにプロトコルを設計する。証明者が解を表すカードやコインを置く（入力する）際は、ゼロ知識性を満たすために検証者から見えない何らかのプライベートな状況にいる必要がある。そこで解を知らない証明者はこの状況を逆手にとり、任意の重さのコインを盤面に置く不正ができてしまう。これを考慮する必要があるため、合計重量のみの比較では健全性を保証できず、従来は 9 枚のコインを 1 枚ずつ比較していた。Ono らのカードベースな方式 [11] のように入力時に工夫することで、入力されたコインの重さが 9 種類しかないことを保証することができればさらに効率化が望める。また Shimano ら [8] は、入力カード列に意図的に傷などのマークが付けられている状況に関する安全性を研究している。これらの検討は将来的な課題とする。

1.4 関連研究

児童の論理的思考力や数学的思考力の育成を目的とした教材として、くもん出版による教材てんびん論理パズルが存在する[*1]。これは、天秤と複数の重さの異なるおもりを用いて、定められた課題に対して重さの比較結果のみを手掛かりとして正解を導出するものである。収録課題は、左右が釣り合うようにおもりを選択する問題や限られた回数の計量で特定の重い（あるいは軽い）おもりを同定する偽コイン問題 [2] のような古典的な課題まで幅広く構成されている。課題は、つり合いの確認や重い対象の識別といった操作を通じて、比較のみによる情報収集と推論を促すよう設計されており、論理的思考や数学的思考の育成を目的として活用されている。

2. 準備

本節では、天秤ベースゼロ知識証明の計算モデルとプロトコルで用いられる 2 種類のソートを説明する。これらのソートは、与えられた複数枚のコインの重さが全て異なることと異なるコイン列同士が等しいことを示すために用いられる。

2.1 計算モデル

天秤ベースゼロ知識証明では、天秤とコインを用いて、コインの重さを比較することで検証する。本節では、使用する天秤およびコインの記法と、その利用方法について述べる。

本プロトコルでは、一般的な上皿天秤を使用する。天秤は、左右の皿に置かれた物体の重量に応じて、重い側に傾

[*1] くもん出版『てんびん論理パズル』、2024 年 7 月。https://shop.kumonshuppan.com/view/item/000000003570?category_page_id=ct80

表 1: コイン枚数と比較回数

プロトコル	比較回数（回）	コイン枚数（枚）
既存プロトコル [4] の拡張	$2\sum_{i\in[1,n]}\ell_i + q(n) + (3n-1)\sum_{i\in[1,n]}(\lfloor\log i\rfloor+1)$	$\sum_{i\in[1,n]}3\ell_i + n - \ell_1$
提案プロトコルの拡張	$2\sum_{i\in[1,n]}\ell_i + n\sum_{i\in[1,n]}(\lfloor\log i\rfloor+1) + 2n$	$\sum_{i\in[1,n]}3\ell_i + n$

く。ただし通常の天秤とは異なり、傾きの速度や角度は重量差に依存せず一定であると仮定する。すなわち、天秤の傾きから得られる情報は、左右どちらの皿の総重量が大きいかだけであり、重量差については一切の情報が得られない。使用するコインは、任意の異なる重さを持つが、外見からは重さを識別できないものとする。

以上を踏まえ、理論的な定義を与える。重さ $w \in \mathbb{R}_{>0}$ をもつ抽象的対象を $c(w)$ とし、これを**コイン**と呼ぶ。コインは視覚的には ◯ などで表されるが、理論上は重み w のみを本質的に有する抽象的対象である。全てのコインから成る集合を C とする。

$$C \coloneqq \{c(w) \mid w \in \mathbb{R}_{>0}\}$$

これらに対して**比較**と**シャッフル**を定義する。

比較：2 つの有限なコイン列 $L = (c(w_1), \ldots, c(w_\ell)) \in C^\ell$ と $R = (c(w'_1), \ldots, c(w'_r)) \in C^r$ に対して、天秤による比較は次のように表される。

$$L \mid R \to \begin{cases} L > R & \text{if } \sum_{i=1}^{\ell} w_i > \sum_{i=1}^{r} w'_i \\ L < R & \text{if } \sum_{i=1}^{\ell} w_i < \sum_{i=1}^{r} w'_i \\ L = R & \text{otherwise} \end{cases}$$

ここでは記号 “|” によって比較を表している。この記号は天秤の支柱を象徴しており、L, R を皿に載せて比較する状況を視覚的に示している。他の表記としては、比較操作を次のような関数として定義することも可能である。

$$\mathsf{compare} : C^* \times C^* \to \{\mathsf{Left}, \mathsf{Right}, \mathsf{Even}\}$$

このような関数形式の記述の方が定式化としては自然であるとも考えられるが、本稿では、視覚的にも天秤の操作の印象を残すことを重視している。

シャッフル：ℓ 枚のコイン列に対するシャッフル操作は、ℓ 次の対称群 S_ℓ に属する置換 π を一様ランダムに選び、それを列に適用する操作である。コイン列 $R = (c_1, \ldots, c_\ell) \in C^\ell$ に対して、シャッフルを次のように表す。

$$\mathsf{shuf}(R) \coloneqq (c_{\pi(1)}, c_{\pi(2)}, \ldots c_{\pi(\ell)})$$

この操作は、コインを手でかき混ぜるようにシャッフルすることで容易に実装できる。

2.2 クイックソート

クイックソート [5] の疑似コード Quicksort をアルゴリズム 1 に示す。これは、ℓ 枚のコイン列 $R \in C^\ell$ に含まれる各コインの重さが全て異なることを、まさに**クイックソート**によって確認するアルゴリズムである。

アルゴリズム 1 クイックソート [5]

```
1:  function Quicksort(R, p, q)
2:      if p ≥ q then
3:          return
4:      end if
5:      i ← p
6:      for j ∈ [p + 1, q] do
7:          if R[p] | R[j] → R[p] = R[j] then
8:              V は動作を終了する
9:          else if R[p] | R[j] → R[p] > R[j] then
10:             R[i + 1] と R[j] を交換
11:             i ← i + 1
12:         end if
13:     end for
14:     R[i] と R[p] を交換
15:     Quicksort(R, p, i − 1)
16:     Quicksort(R, i + 1, q)
17:     return
18: end function
```

$$\textsc{Quicksort}(R, 0, \ell-1) = \begin{cases} 1 & \text{if } w_i \neq w_j \text{ for any } i, j \in [1, \ell] \\ 0 & \text{otherwise} \end{cases}$$

ただし $R = (c(w_1), \ldots, c(w_\ell))$ であり、$[1, \ell]$ は整数区間を表す。アルゴリズム 1 と一般的なクイックソートとの違いは 8 行目に現れており、ピボット $R[p]$（p 番目のコインを示す）との比較結果が等しい場合は、動作を止めることである。これは、比較結果が等しい場合は、R に同じ重さのコインが 2 枚含まれることを意味するためである。

2.3 クイックソート*

クイックソート*の疑似コード Quicksort*をアルゴリズム 2 に示す。これは、すでにソートされたコイン列 $R' \in C^\ell$ からピボットとなるコインを選択することで、常に最適なピボットを使って $R \in C^\ell$ をクイックソートする。その過程で、ソート済みの R' とコイン列 R が順序を無視して一致すること（本稿ではこれを $R = R'$ と表記する）を確認する。

$$\textsc{Quicksort}^*(R', R, 0, \ell-1) = \begin{cases} 1 & \text{if } R = R' \\ 0 & \text{otherwise} \end{cases}$$

ここで R' はアルゴリズム 1 で事前にソートされていたとすると、R' に含まれる各コインの重さは異なるため、アルゴリズム 2 を行うと R に含まれる各コインの重さも異なる

アルゴリズム 2 クイックソート* [5]

```
 1: function QUICKSORT*(R, R′, p, q)
 2:     if p ≥ q then
 3:         if R[p] | R′[p] → R[p] = R′[p] then
 4:             return
 5:         end if
 6:         V は動作を終了する
 7:     end if
 8:     (i, m, isChecked) ← (p, ⌈(p + q)/2⌉, 0)
 9:     for j ∈ [p, q] do
10:         if R[j] | R′[m] → R[j] = R′[m] then
11:             if isChecked = 1 then
12:                 V は動作を終了する
13:             end if
14:             R[i] と R[j] を交換
15:             (i′, i, isChecked) ← (i, i + 1, 1)
16:         else if R[j] | R′[m] → R[j] > R′[m] then
17:             R[i] と R[j] を交換
18:             i ← i + 1
19:         end if
20:     end for
21:     if isChecked = 0 then
22:         V は動作を終了する
23:     end if
24:     R[i′] と R[m] を交換
25:     QUICKSORT*(R, R′, p, m − 1)
26:     QUICKSORT*(R, R′, m + 1, q)
27:     return
28: end function
```

ことが言える。

アルゴリズム 2 では R' から常に最適なピボット $R'[m]$ を取っているため、アルゴリズム 1 と異なり、R の中で $R'[m]$ と等しいコインが判明（10 行目）すると、そのコインの位置を記録（15 行目の $i' \leftarrow i$）し、最後にそのコインが真ん中となるように入れ替えている（24 行目）。ピボットと等しいコインが初めて判明したことをフラグにセットし（15 行目の isChecked ← 1）、もしも再度ピボットと等しいコインが現れた場合（11 行目）やフラグが 0 のままである場合（21 行目）は、$R \neq R'$ が判明するため動作を止める。

3. 既存プロトコルの拡張

本節では、著者らが提案した既存プロトコル [4] を紹介し、一般化された $n \times n$ の数独に対する疑似コードを与える。

3.1 既存プロトコル

数独の解に対するゼロ知識証明では、与えられた数独の問題に対して各行/列/ブロックに 1 から 9 までの数字が全て一度ずつ出現するような数字の配置が存在することを検証する。既存プロトコルの流れは次の通りである。準備として、証明者 P は、数字に対応する重さのコインを各マスに 3 枚置くことで、自身の持つ解をコインの重さで表す。P は、1 行目に対応する 9 マスからコインを 1 枚ずつ取り、9 枚のコインの重さが全て異なることをアルゴリズム 1 によって V に示す。次に他の各行/列/ブロックに対応する 9 マスからコインを 1 枚ずつ取り、1 行目に置かれた 9 枚のコインと一致すること、すなわち 9 枚の異なるコインが置かれていることをアルゴリズム 2 によって V に示す。このとき、事前に数字が書かれた各マス（以降では**ヒントマス**と呼ぶ）にコインを置かないことでコインの枚数と比較回数を削減する [6]。

3.1.1 準備

証明者 P は、次の手順に従って、自身が持つ数独の解を表現する。すなわち、各マスに、そのマスの数字に対応する重さのコインを配置する。ここで、各数字 $i \in [1, 9]$ に対して対応関係 $w_i \in \mathbb{R}_{>0}$ を定め、各 $i < j$ に対して $w_i < w_j$ を満たすとする。ただし、P が不正を行う可能性にも留意する必要があり、解を知らない P が、V の監視の届かない範囲で、本来対応しない重さのコインを反則的に配置してしまう恐れがあることに注意されたい。

(1) V は 1 行目の各ヒントマスに対して、対応する重さのコインを 1 枚置く。

(2) P は各空きマスに対して、自身の持つ解に従い、対応する重さのコインを 3 枚置く。

(3) P は各空きマスに置かれた 3 枚のコイン c_1, c_2, c_3 に対して、重さが等しいことを次のように V に示す。

$$c_1 \mid c_2 \ \&\ c_1 \mid c_3 \quad \rightarrow \quad c_1 = c_2 \ \&\ c_1 = c_3$$

ここで天秤が 1 回でも傾いた場合、V は以降の操作を打ち切り、検証を終了する。

この準備によって、各空きマスには重さが等しい 3 枚のコインが置かれていることが保証される。1 行目の各ヒントマスにのみコインが 1 枚置かれているが、そのほかのヒントマスにはコインは置かれていない。

3.1.2 検証

盤面に置かれたコインが数独のルールを満たすことを、次のように確認する。

(1) P は 1 行目の 9 つのマスからコインを 1 枚ずつ取り、それらをシャッフルする。これら 9 枚のコイン列を R_1 と表す。

(2) R_1 に対してアルゴリズム 1 を適用する。

(3) P は他の各 $i \in \{2, 27]$ 行目の空きマスからコインを 1 枚ずつ取り、それらをシャッフルする。ただし $10 \leq i \leq 18$ は列、$19 \leq i \leq 27$ はブロックに対応する。これらコイン列を R_i とする。

(4) R_1 を用いて、各 R_i に対してアルゴリズム 2 を適用する。ここで i 行目のヒントマスを考慮し、R_1 からそのヒントマスに対応するコインを除いてアルゴリズム 2 を実行する。

全ての操作を実行できると、1 行目に配置された 9 つの数字が全て異なり、他の各行/列/ブロックにも同じ 9 つの異なる数字が含まれていることを検証できる。つまり、V

は P が解を持っていることを納得する。

3.2 $n \times n$ の一般化数独に対する拡張

既存プロトコルの記述を $n \times n$ の一般化数独に対して拡張する。疑似コードをアルゴリズム 3 に示す。ここで盤面の各マスを座標として表し、(i, j) を i 行 j 行目のマスとする。ヒントマスを考慮するため、P が置くコインの他に入力として $H_{i,j} \in \{0, 1\}$ を追加し、マス (i, j) が空きマスか否かを表す。各コイン $c_{(i,j,k)}$ はマス (i, j) に対応するコインを表し、$k \in [1, 3]$ は 1 つのマスに置かれた 3 枚のコインを区別する。ただし $H_{i,j} = 1$ のとき、すなわちマス (i, j) がヒントマスのときはコイン $c_{(i,j,k)}$ は置かれず、入力として定義しないが、1 行目のヒントマスにはコインが 1 枚置かれる。これにより各コイン列 R_i は、R_1 は n 枚となり、他の R_i は対応するヒントマスが考慮された枚数になる。

検証ではまず、空きマスに置かれた 3 枚のコインが同一であることの確認（アルゴリズム 3 の 1 行目から 10 行目まで）や各コイン列のシャッフル（11 行目から 13 行目）および R_1 に対して Quicksort を適用（14 行目）する。Quicksort*の適用（15 行目以降）においては、ソートされた R_1 からヒントマスを考慮してコインを抜き出し、新たに R'_1 を構成している。

3.3 性能評価

$i \in [1, n]$ 行目の空きマスの数を ℓ_i とすると、次が成り立つ。

$$\ell_i = n - \sum_{j \in [1,n]} H_{i,j}$$

全ての空きマスにコインを 3 枚ずつ置き、1 行目のヒントマス（$n - \ell_1$ 個）にはコインを 1 枚ずつ置くため、必要なコインの枚数は合計 $\sum_{i \in [1,n]} 3\ell_i + n - \ell_1$ 枚となる。

比較回数は、まず準備で行った比較に加えて、1 行目に対する 1 回のアルゴリズム 1 と他の行/列/ブロックに対する $3n - 1$ 回のアルゴリズム 2 に必要な比較回数の合計となる。アルゴリズム 1 に必要な比較回数はコインの順番によって確率的となるため、これを $q(n)$ とする。ただし $q(n) = O(n \log n)$ である。アルゴリズム 2 に必要な比較回数を $q^*(n)$ とすると、これは以下の漸化式で与えられる。

$$q^*(n) = n + q^*\left(\left\lfloor \tfrac{n-1}{2} \right\rfloor\right) + q^*\left(\left\lceil \tfrac{n-1}{2} \right\rceil\right), \quad q^*(0) = 0$$

これを計算すると $q^*(n) = \sum_{i \in [1,n]} (\lfloor \log i \rfloor + 1)$ を得る。したがって比較回数の合計は次の通りである。

$$2 \sum_{i \in [1,n]} \ell_i + q(n) + (3n - 1) \sum_{i \in [1,n]} (\lfloor \log i \rfloor + 1) \qquad (1)$$

4. 合計を利用した効率化

本節では、コインを 1 枚ずつ検証する既存プロトコル [4] に対して、コインを重ねてコインの合計を比較することで効率化する手法を提案する。

アルゴリズム 3 一般化数独に対する既存プロトコル [6]

Require: $H_{i,j} \in \{0, 1\}$ for all $i, j \in [1, n]$
Require: $c_{(i,j,k)} \in C$ for all $i, j \in [1, n]$, $k \in [1, 3]$ with $H_{i,j} = 0$
Require: $c_{(1,j,1)} \in C$ for all $j \in [1, n]$ with $H_{1,j} = 1$
Require: $R_i \leftarrow (c_{(i,1,1)}, c_{(i,2,1)}, \ldots, c_{(i,n,1)})$ for all $i \in [1, n]$
Require: $R_{n+j} \leftarrow (c_{(1,j,2)}, c_{(2,j,2)}, \ldots, c_{(n,j,2)})$ for all $j \in [1, n]$
Require: $R_{2n+\sqrt{n}(p-1)+q} \leftarrow (c_{(\sqrt{n}p-\sqrt{n}+1, \sqrt{n}q-\sqrt{n}+1, 3)}, \ldots, c_{(\sqrt{n}p, \sqrt{n}q, 3)})$ for all $p, q \in [1, \sqrt{n}]$
Ensure: $c_{(i,j,1)} = c_{(i,j,2)} = c_{(i,j,3)}$ for all $i, j \in [1, n]$ with $H_{i,j} = 0$
Ensure: $R_k[i] \neq R_k[j]$ for all $i, j \in [1, |R_k|]$ and $k \in [1, 3n]$ with $i < j$
1: **for** $i, j \in [1, n]$ **do**
2: **if** $H_{i,j} = 0$ **then**
3: **if** $c_{(i,j,1)} \mid c_{(i,j,2)} \rightarrow c_{(i,j,1)} \neq c_{(i,j,2)}$ **then**
4: V は動作を終了する
5: **end if**
6: **if** $c_{(i,j,2)} \mid c_{(i,j,3)} \rightarrow c_{(i,j,2)} \neq c_{(i,j,3)}$ **then**
7: V は動作を終了する
8: **end if**
9: **end if**
10: **end for**
11: **for** $i \in [1, 3n]$ **do**
12: $R_i \leftarrow \mathsf{shuf}(R_i)$
13: **end for**
14: Quicksort$(R_1, 0, n - 1)$
15: **for** $i \in [2, 3n]$ **do**
16: **if** $2 \le i \le n$ **then**
17: $c'_{(i,j)} \leftarrow R_1[j]$ for all $j \in [1, n]$ with $H_{i,j} = 0$
18: **else if** $n + 1 \le i \le 2n$ **then**
19: $c'_{(i,j)} \leftarrow R_1[j]$ for all $j \in [1, n]$ with $H_{j,i-n} = 0$
20: **else if** $2n + 1 \le i \le 3n$ **then**
21: $b \leftarrow i - 2n$
22: $p \leftarrow \lceil \frac{b}{\sqrt{n}} \rceil$, $q \leftarrow b - \sqrt{n}(p - 1)$
23: $c'_{(i,(u-\sqrt{n}(p-1)-1)\sqrt{n}+(v-\sqrt{n}(q-1)))} \leftarrow R_1[(u - \sqrt{n}(p-1)-1)\sqrt{n}+(v-\sqrt{n}(q-1))]$ for all $u \in [\sqrt{n}p - \sqrt{n}+1, \sqrt{n}p]$, $v \in [\sqrt{n}q - \sqrt{n}+1, \sqrt{n}q]$ with $H_{u,v} = 0$
24: **end if**
25: $R'_1 \leftarrow (c'_{(i,1)}, c'_{(i,2)}, \ldots, c'_{(i,n)})$
26: Quicksort*$(R'_1, R_i, 0, |R_i| - 1)$
27: **end for**

4.1 アイデア

既存プロトコルの検証を観察すると、次の事実が得られる。全ての行（ここでは 1 行目から n 行目の意味）の検証が完了すると、数独の全てのマスで 1 から n までの数字がちょうど n 回ずつ出現したことが保証される。準備の段階で各マスに置かれた 3 枚のコインは等しいことが保証されているため、以降の検証で扱う各列/ブロックには、n つの重さのコインしか存在しないことが言える。

以上の事実を利用し、各列/ブロックに含まれるコインの重さの合計のみを検証する方法が考えられる。すなわち合計重量が $W = \sum_{i \in [1,n]} w_i$ に一致するかどうか比較することで、各列/ブロックの検証を 1 回の比較のみで済ませる発想が得られる。しかしながら、この方法でもなお健全性に問題がある。すなわち、同じ重さのコインが複数含まれても合計が W になるように不正に配置した可能性を排除できない。これは、コインの合計重量が W に一致したとして

も、それが必ずしも異なる重さのコインを含んでいることを意味しないためであり、単に合計のみを比較する方法では、検証者が誤った解に納得してしまう可能性がある。

この健全性の問題を回避する一案として、コインの重さに特殊な値を割り当てる方法が考えられる。例えば $w_i = 2^{i-1}$ のように 2 のべき乗を割り当てると，二進表記の一意性より，合計のみを確かめることでコインの一意性が保証される．これによって、R_1 から R_n のコイン列（行に対応）はそれぞれシャッフルする必要があるが、R_{n+1} 以降（列とブロックに対応）は重さの合計値のみ確認するためシャッフルする必要は無く、シャッフル回数の削減も見込まれる。ただし $w_i = 2^{i-1}$ であることを保証するためには、R_1 のソート後に各コインを重さが 2^{i-1} のコインと比較する必要がある。また、2 のべき乗を用いるとコインの重さが指数的に大きくなり、これも問題である。他の数の利用について後ろの節で詳しく議論する。

4.2 提案プロトコル

既存プロトコルと同様に、各空きマスに 3 枚コインを置き、各マスに置かれた 3 枚のコインが等しいことを確認する。ここで、f をある関数とし、各 $i \in [1,n]$ に対してコインの重さを $w_i = f(i)$ と定める。本プロトコルでは、このような対応関係を満たしていて、予め重さが分かっている n 枚のコイン $(c(w_1), \ldots, c(w_n))$ を追加で用意する。

提案プロトコルは以下の手順で検証を行う。プロトコルの疑似コードをアルゴリズム 4 に示す。

(1) 3.1.2 節のステップ 3 と同様の手順で P は各 $i \in [1,n]$ 行目のコインをシャッフルし、コイン列 R_i を作成する。
(2) あらかじめ用意したコインを用いて 3.1.2 節の (4) と同様の手順で R_1 から R_n があらかじめ用意したコイン列と等しいことの検証を行う。
(3) 次に各列/ブロックの n 個の各マスからコインを 1 枚取り、コイン列とする。
(4) 上で作成したコイン列と R_1 のコイン列を列にしたものを比較し、全てのコイン列の重さがそれぞれ R_1 のコインの重さの合計と等しくなることを検証する。

4.3 性能評価

全ての空きマスに 3 枚ずつコインを置き、コイン列 R_0 を準備するため、必要なコインの枚数は合計 $\sum_{i\in[1,n]} 3\ell_i + n$ 枚となる。比較回数は、n 回の最適クイックソートに必要な比較回数と各列/ブロックのコイン列と R_0 との比較の合計となる。

したがって比較回数の合計は次の通りである。

$$2\sum_{i\in[1,n]} \ell_i + n\sum_{i\in[1,n]} (\lfloor \log i \rfloor + 1) + 2n \qquad (2)$$

アルゴリズム 4 合計を用いたプロトコル

Require: $H_{i,j} \in \{0,1\}$ for all $i, j \in [1,n]$
Require: $R_0 \leftarrow (c_1, \ldots, c_i)$ for all $i \in [1,n]$
Require: $c_{(i,j,k)} \in C$ for all $i, j \in [1,n]$, $k \in [1,3]$ if $H_{i,j} = 0$
Require: $R_i \leftarrow (c_{(i,1,1)}, c_{(i,2,1)}, \ldots, c_{(i,n,1)})$ for all $i \in [1,n]$
Require: $R_{n+j} \leftarrow (c_{(1,j,2)}, c_{(2,j,2)}, \ldots, c_{(n,j,2)})$ for all $j \in [1,n]$
Require: $R_{2n+\sqrt{n}(p-1)+q} \leftarrow (c_{(\sqrt{n}p-\sqrt{n}+1, \sqrt{n}q-\sqrt{n}+1, 3)}, c_{(\sqrt{n}p-\sqrt{n}+1, \sqrt{n}q-\sqrt{n}+2, 3)}, \ldots, c_{(3p,3q,3)})$ for all $p, q \in [1, \sqrt{n}]$
Ensure: $c_{(i,j,1)} = c_{(i,j,2)} = c_{(i,j,3)}$ for all i, j with $H_{i,j} = 0$
Ensure: $R_k[i] \neq R_k[j]$ for all $i, j \in [1, |R_k|]$ and $k \in [1, 3n]$ with $i < j$

for $i, j \in [1,n]$ **do**
  if $H_{i,j} = 0$ **then**
    if $c_{(i,j,1)} \mid c_{(i,j,2)} \rightarrow c_{(i,j,1)} \neq c_{(i,j,2)}$ **then**
      V は動作を終了する
    end if
    if $c_{(i,j,2)} \mid c_{(i,j,3)} \rightarrow c_{(i,j,2)} \neq c_{(i,j,3)}$ **then**
      V は動作を終了する
    end if
  end if
end for
for $i \in [1,n]$ **do**
  shuf(R_i)
end for
for $i \in [1,n]$ **do**
  $c'_{(i,j)} \leftarrow R_1[j]$ for all $j \in [1,n]$ with $H_{i,j} = 0$
  $R'_1 \leftarrow (c_{(i,1)}, \ldots, c_{(i,n)})$
  Quicksort*$(R_0, R_i, 0, |R_i| - 1)$
end for
for $i \in [n+1, 3n]$ **do**
  if $n+1 \le i \le 2n$ **then**
    $c'_{(i,j)} \leftarrow R_0[j]$ for all $j \in [1,n]$ with $H_{j,i-n} = 0$
  else if $2n+1 \le i \le 3n$ **then**
    $b \leftarrow i - 2n$
    $p \leftarrow \lceil \frac{b}{\sqrt{n}} \rceil$, $q \leftarrow b - \sqrt{n}(p-1)$
    $c'_{(i,(u-\sqrt{n}(p-1)-1)\sqrt{n}+(v-\sqrt{n}(q-1)))} \leftarrow R_1[(u-\sqrt{n}(p-1)-1)\sqrt{n}+(v-\sqrt{n}(q-1))]$ for all $u \in [\sqrt{n}p-\sqrt{n}+1, \sqrt{n}p]$, $v \in [\sqrt{n}q-\sqrt{n}+1, \sqrt{n}q]$ with $H_{u,v} = 0$
  end if
  $R'_0 \leftarrow (c_{(i,1)}, \ldots, c_{(i,n)})$
  if $R'_0 \mid R_i \rightarrow R'_0 \neq R_i$ **then**
    V は動作を終了する
  end if
end for

5. 考察

本節では、提案プロトコルで用いるコインの重さについて考察する。まず，各コイン $c(w_i) \in C$ の重さ w_i に，単に連続する整数を割り当てると，提案プロトコルのように合計のみの比較では健全性エラーが生じることを詳しく述べる．

5.1 連続する整数

合計を用いるプロトコルにおいて、コインの重量が連続する整数値で割り当てられている場合、特定の条件下で検証を通過できる不正な配置を構成可能である。具体的に

は、同一ブロック内で和が等しい異なる 2 組の数字を考える。例えば $w_i \in [1,n]$ とすると、$(c(1),c(4))$ と $(c(2),c(3))$ のように各重さは異なるが合計重量は等しい組が 2 つ存在する。このうち一方の組を座標 (i,j) と $(i+1,j)$ に縦に配置し、もう一方の組を $(i,j+1)$ と $(i+1,j+1)$ に縦に配置する。すなわち、隣接する列にそれぞれ縦方向に並べる場合を考える。このとき、(i,j) と $(i,j+1)$ を入れ替え、$(i+1,j)$ と $(i+1,j+1)$ を入れ替えても、ブロック全体および列全体の合計重量は不変である。さらにこれは横方向の入れ替えであるため、行内に含まれるコインの組み合わせは変化せず、行の検証も通過する。すなわち、もともとは数独のルールに従って構成された盤面から、上記の横方向の入れ替えを施すことで、同じ列に同じ数字が含まれるという数独のルールに反する盤面を作り出せる。それにもかかわらず、各列/ブロックの合計検証は通過するため、プロトコルはこの不正な盤面を正しい解として受理し、問題となる。

5.2 特殊な数列

上の問題に対処するためには、コインの重さの割り当て方法を工夫し、n 種類のコインから重複ありで n 個のコインを選んだ際の合計重量が、各種類を 1 枚ずつ選んだ場合にのみ特定の値 W となるような重さであればよい。すなわち、n 個のコインから成る集合

$$C = \{c(w_i) \in C \mid 1 \le i \le n, w_j \ne w_k, 1 \le j < k \le n\}$$

に対して、各 $c(w_i)$ の選び方を $k_i \in [0,n]$ とすると、$\sum_{i=1}^{n} k_i = n$ である。このとき次が成り立てばよい.

$$\sum_{i=1}^{n} k_i w_i = W \iff k_1 = k_2 = \cdots = k_n = 1$$

例えば上で議論したように、2 のべき乗はこの性質を満たすが、$n \times n$ の一般化数独において $n = 16$ の場合、最大重量は $2^{16} - 1 = 65,535$ となり、実際の物理コインとして扱うには重量差が極端である。

この解決策として、分数や小数をコインの重さに利用する方法が考えられる。例えば、分母を互いに素に設定した分数列 $\left\{\frac{1}{p_i} \mid p_i \text{ は } i \text{ 番目の素数}\right\}$ や $\frac{1}{2^i}$ のような等比数列を用いれば、合計が W になるコインの選び方が一意であることを保ったまま総重量の増大を抑えることができる。また、小数に関しては、大きくなりすぎた数列全体の桁を小数点でスライドさせることで、最大重量を現実的な範囲に収めることができる。例えば、2 のべき乗列を $\frac{1}{M}$ 倍にスケーリングして $\left\{\frac{1}{M} \cdot 2^i \,\middle|\, i \in \mathbb{N}\right\}$ とすれば、重量差はそのまま維持されつつ、現実的な重量差に抑えることができる。この場合も合計が W になるコインの選び方が一意であることは損なわれないため、検証の健全性は保証される。

5.3 分配の方法が一意になるコインの重さ

提案プロトコルで用いるコインの重さの割り当ては、特定の条件下で一意に分配が決まる数列を選択する問題として次のように定式化できる。参加者数を n 人とし、各参加者は n 枚のコインを受け取るものとする。ここでコインの重さは n 種類の値をもち、各重さのコインが n 枚ずつあるものとする。各参加者の受け取るコインの重さの合計が n 種類のコインを一枚ずつ受け取った時のものとなるようにコインを配る時、各参加者が受け取るコインの組み合わせが一意に決定されるような数列を選択する。すなわち、合計が W となる異なる組み合わせが存在しないように、コインの重さを決定する。この一意性が保証されれば、検証において列やブロックについて合計重量のみを比較する方式であっても、その正当性を確実に確認でき、不正な入れ替え操作によって検証を突破されることを防止できる。この問題を満たす数列の探索は将来的な課題とする。

謝辞 本研究の一部は JSPS 科研費 JP23H00479 の助成及び JSPS 二国間交流事業 JPJSBP120253206 の支援を受けている。また九州大学マス・フォア・インダストリ研究所共同利用・共同研究拠点の支援を受けた。(2025 年度研究集会「産学連携と数理・暗号分野連携によるカードベース暗号の深化と新境地 II」(2025a036))

参考文献

[1] Aumann, Y. and Lindell, Y.: Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries, *Theory of Cryptography* (Vadhan, S. P., ed.), LNCS, Vol. 4392, Berlin, Heidelberg, Springer, pp. 137–156 (2007).

[2] E.D.Schell: Problem E651-Weighed and found wanting, *Amer. Math. Monthly*, Vol. 52, p. 42 (1945).

[3] Gradwohl, R., Naor, M., Pinkas, B. and Rothblum, G. N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles, *Theory of Computing Systems*, Vol. 44, No. 2, pp. 245–268 (2009).

[4] Kaneko, S., Lafourcade, P., Mallordy, L.-B., Miyahara, D., Puys, M. and Sakiyama, K.: Balance-Based ZKP Protocols for Pencil-and-Paper Puzzles, *Information Security* (Mouha, N. and Nikiforakis, N., eds.), LNCS, Vol. 15257, Cham, Springer, pp. 211–231 (2025).

[5] Kaneko, S., Sakiyama, K. and Miyahara, D.: Development on Balance-Based Zero-Knowledge Proof Syestems, SCIS 2025, 4D1-1 (2025).

[6] Kaneko, S., Sakiyama, K. and Miyahara, D.: Efficient Balance-Based ZKP Protocol for Sudoku, *IEICE Tech. Rep.*, ISEC2025-66, Vol. 125, No. 99, pp. 332–339 (2025).

[7] Ruangwises, S.: Two standard decks of playing cards are sufficient for a ZKP for Sudoku, *New Gener. Comput.*, Vol. 40, No. 1, pp. 49–65 (2022).

[8] Shimano, M., Sakiyama, K. and Miyahara, D.: Towards Verifying Physical Assumption in Card-Based Cryptography, *Innovative Security Solutions for Information Technology and Communications* (Bella, G., Doinea, M. and Janicke, H., eds.), LNCS, Vol. 13809, Cham, Springer, pp. 289–305 (2023).

[9] Tanaka, K. and Mizuki, T.: Two UNO Decks Efficiently Perform Zero-Knowledge Proof for Sudoku, *Fundamentals of Computation Theory* (Fernau, H. and Jansen, K., eds.), LNCS, Vol. 14292, Cham, Springer, pp. 406–420 (2023).

[10] Tanaka, K., Sasaki, S., Shinagawa, K. and Mizuki, T.: *Only*

Two Shuffles Perform Card-Based Zero-Knowledge Proof for Sudoku of Any Size, pp. 94–107, ACM (2025).

[11] Tomoki, O., Suthee, R., Yoshiki, A., Kyosuke, H. and Mitsugu, I.: Single-Shuffle Physical Zero-Knowledge Proof for Sudoku Using Interactive Inputs, *ACM Asia Public-Key Cryptography Workshop*, New York, ACM, pp. 1–8 (2025).

[12] Yato, T. and Seta, T.: Complexity and completeness of finding another solution and its application to puzzles, *IEICE Trans. Fundamentals*, Vol. 86, No. 5, pp. 1052–1060 (2003).