A Bit of a Puzzle

Daniel Rosina

April 13, 2026

1 The Puzzle

Oh no! You and your friend have been locked up in mathematical jail for claiming that 0 is a natural number. The strangely maths-obsessed warden offers you a chance to escape, but only if the two of you can solve the following puzzle: you walk into a room to find an 8 by 8 chessboard, with a coin placed on every tile, each set by the warden to be either heads or tails. The warden hides a key under one of those 64 tiles and allows you to see exactly which one. You are then permitted to flip exactly one coin, leaving the rest of the board untouched. Your prisoner friend then walks into the room and, without any prior knowledge or external communication, must deduce from the resulting configuration of coins the exact tile concealing the key. How might the two of you devise a strategy that guarantees your freedom?

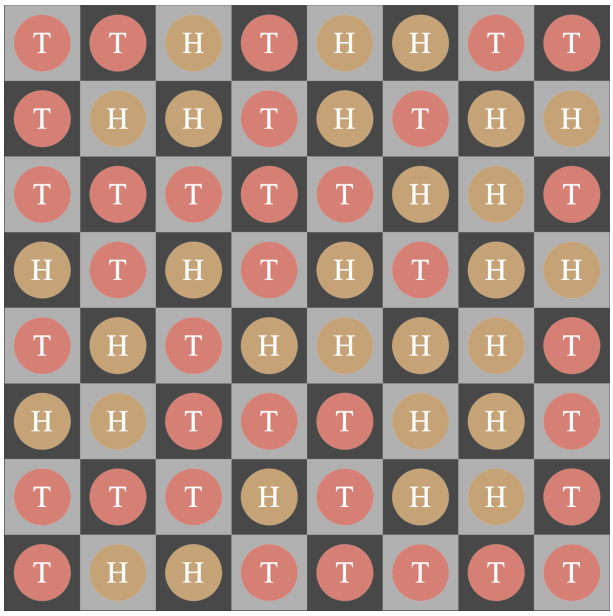

Figure 1: The Original Puzzle

I first encountered this puzzle in a Matt Parker video five years ago [1], and it has captured my attention since, both for its elegant solution and its further generalisations. So, dear reader, if you have not seen it before, I highly encourage you to try it for yourself first. That euphoric feeling of cracking a puzzle on your own can only happen once.

Are we ready? Let's go!

2 Intuition

I imagine many people share my initial gut reaction: to immediately blurt out 'This is impossible!' After all, it seems extraordinary that, from any of the 2^{64} possible starting configurations of coins, a single flip could always allow our friend to identify the correct

tile. But before we get ahead of ourselves, let us formulate the problem a little more precisely.

2.1 Definitions

At its core, this problem is one of finding an *encoding function* that, given a particular coin configuration, identifies the tile under which the key is hidden. Let us label the tiles 0 to $n-1$[1], and let 1 denote Heads while 0 denotes Tails. We can then represent any configuration of the board as an n-dimensional vector

$$a = (a_0, a_1, \ldots, a_{n-1}) \in \{0,1\}^n \quad \text{where } a_i \in \{0,1\} \ \ \forall i, \tag{1}$$

where a_i denotes the value of the coin on tile i.[2]

We then define the encoding function that maps such a configuration to a specific tile index:

$$E : \{0,1\}^n \to \mathbb{Z}, \quad E(a) = m \quad \text{where } m \in \{0,1,\ldots,n-1\}. \tag{2}$$

We want this function to have the following key property: for any arbitrary $a \in \{0,1\}^n$, we can flip a single entry of a so that E encodes any desired value in $\{0,1,\ldots,n-1\}$. Determining when such a function exists in general is the central question of this essay, and for the original puzzle we investigate $n = 64$.

2.2 2 coins

As any good mathematician will tell you, the best way to approach a difficult problem is to first look at simpler cases. The simplest non-trivial case is $n = 2$: two coins on two tiles[3]. After experimenting with a few different encodings, I found that a natural strategy is to let coin 1 encode the tile directly: if it is heads, the key is under tile 1; if it is tails, the key is under tile 0. This gives the encoding function

$$E(a) = a_1.$$

A quick check confirms that this strategy works in every case[4].

2.3 4 coins

Moving to the 2 by 2 (4 coin) case, it quickly becomes clear that this simple strategy will not scale. Back to the drawing board.

[1] I apologise to all the 1-indexing fans out there, but your view is objectively wrong.

[2] As you will see, the geometric arrangement of the coins is irrelevant to the solution – as long as you and your partner agree on a consistent ordering of the tiles, the argument remains identical.

[3] The $n = 1$ case is trivial, as the key is always under the single tile.

[4] In some versions of this problem you are required to flip exactly 1 coin and are not permitted to leave the board unchanged – in this essay we will consider both variants.

After some head-scratching, I tried modular arithmetic: since the target tile is an integer, defining E as a weighted sum taken modulo n felt like a natural way to reach any value in $\{0, \ldots, n-1\}$ with a single adjustment. This gives

$$E(a) = 0 \cdot a_0 + 1 \cdot a_1 + 2 \cdot a_2 + 3 \cdot a_3 \pmod 4 = \sum_{i=0}^{3} i \cdot a_i \pmod 4.$$

This fails for configurations like $a = (0, 1, 0, 0)$ with target 2: the encoding gives $E(a) = 1$, but reaching tile 2 would require flipping coin 1 (already heads) or coin 3 (already tails), neither of which are available. In general, reaching target tile t from encoding m requires one of exactly two coins to be in the right state, giving a 75% success rate, and an adversarial warden could exploit the remaining 25% to guarantee our failure.

Why does this approach break down? The core issue is that flipping a coin from heads to tails and from tails to heads have *asymmetric* effects on E: one adds something modulo n, the other subtracts it. To fix this, I turned to *parity*, since toggling an element either in or out of a set changes its parity identically regardless of direction.

There is also a useful information-theoretic lens here. The fundamental measure of information is

$$I(p) = -\log_2 (p(x)) . \tag{3}$$

Since there are n tile choices, $p(x) = \frac{1}{n}$, and so we need to communicate $\log_2(n)$ bits of information. In the $n = 4$ case, that means conveying exactly 2 bits through our single coin flip.

Each integer from 0 to 3 has a 2-bit binary representation, so the two bits map naturally onto the two digits. A clean way to encode each bit is via the parity of a chosen subset of coins. The scheme I arrived at uses the parity of the first row for the first bit and the parity of the first column for the second:

$$E(a) = 2 \cdot ((a_0 + a_1) \bmod 2) + 1 \cdot ((a_0 + a_2) \bmod 2).$$

As a worked example, take $a = (1, 1, 0, 1)$, with the key under tile 2.

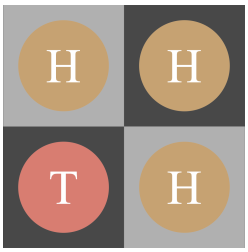

Figure 2: The 4-coin configuration $a = (1, 1, 0, 1)$

The first row has 2 heads (even parity, so bit 1 is 0) and the first column has 1 head (odd parity, so bit 2 is 1), giving $E(a) = 01_2 = 1$. To reach tile $2 \equiv 10_2$, both bits must flip, which corresponds to toggling a_0, giving $a' = (0, 1, 0, 1)$ and $E(a') = 2$.

Crucially, in this parity-based scheme, flipping a coin from heads to tails and from tails to heads have *identical* effects, as both toggle the parity. I find it helpful to think of this

as partitioning the tiles into sets $\{a_0, a_1\}, \{a_0, a_2\}$ whose parities encode individual bits. The Venn diagram below illustrates that every combination of set memberships contains at least one tile, guaranteeing that any target encoding is always reachable in exactly one flip by simplifying identifying the bits that need to change.

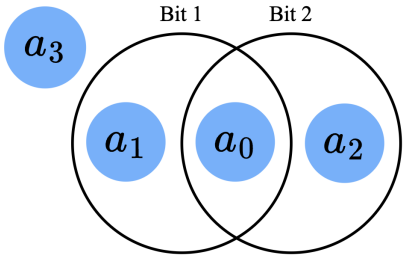

Figure 3: Bit Set Venn Diagram

With this intuition in hand, let us return to the 8 by 8 case.

3 The Solution

Generalising to the 8 by 8 board turns out to be surprisingly straightforward. We now need to convey $\log_2(64) = 6$ bits of information, and each integer from 0 to 63 can be expressed as a 6-digit binary number. The task is simply to find 6 sets analogous to the 2 we used in the 4-coin case.

Rather than assigning numbers to sets arbitrarily, there is a natural construction. Since each integer from 0 to 63 has 6 binary digits, we define one set for each digit position: the ith set S_i consists of all tile indices whose ith binary digit is 1. Applying this for each of the 6 digit positions gives the following six sets of 32 elements each:

S_0							
1	3	5	7	9	11	13	15
17	19	21	23	25	27	29	31
33	35	37	39	41	43	45	47
49	51	53	55	57	59	61	63

S_1							
2	3	6	7	10	11	14	15
18	19	22	23	26	27	30	31
34	35	38	39	42	43	46	47
50	51	54	55	58	59	62	63

S_2							
4	5	6	7	12	13	14	15
20	21	22	23	28	29	30	31
36	37	38	39	44	45	46	47
52	53	54	55	60	61	62	63

S_3							
8	9	10	11	12	13	14	15
24	25	26	27	28	29	30	31
40	41	42	43	44	45	46	47
56	57	58	59	60	61	62	63

S_4							
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

S_5							
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Figure 4: The six parity sets S_i for $i \in \{0, 1, 2, 3, 4, 5\}$

We can verify the key property: every possible combination of set memberships corresponds to a unique tile, which is simply equivalent to every 6-bit binary string having a unique corresponding integer, which is clearly true[5]. Hence we assemble all six bits to get the full encoding

$$E(a) = 32 \cdot \left(\sum_{i \in S_5} a_i \pmod 2 \right) + 16 \cdot \left(\sum_{i \in S_4} a_i \pmod 2 \right) + 8 \cdot \left(\sum_{i \in S_3} a_i \pmod 2 \right) + 4 \cdot \left(\sum_{i \in S_2} a_i \pmod 2 \right) + 2 \cdot \left(\sum_{i \in S_1} a_i \pmod 2 \right) + 1 \cdot \left(\sum_{i \in S_0} a_i \pmod 2 \right).$$

Moreover, this construction gives a pleasing interpretation: the parity of set S_i encodes precisely the ith binary digit of the target tile.

Finally, we introduce the last ingredient: the $\oplus$ (XOR) operation. For binary digits, $\oplus$ is defined as:

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 1 &= 0. \end{aligned}$$

This is equivalent to addition modulo 2, and therefore to computing the parity of a collection of bits. Recall that the parity of set S_i, or equivalently the XOR of its elements' coin values, encodes precisely the ith binary digit of our target tile. Concretely, the parity of S_i among the heads-up coins is

$$\bigoplus_{j \in S_i} a_j = \sum_{j \in S_i} a_j \pmod 2,$$

which gives the ith bit of $E(a)$, therefore it can be rewritten as

$$E(a) = \sum_{i=0}^{5} \left(2^i \bigoplus_{j \in S_i} a_j \right).$$

Now here is the key observation: since $\oplus$ operates *independently* on each bit, the ith bit of $p \oplus q$ is simply the ith bit of p XOR'd with the ith bit of q. It follows that taking the bitwise XOR of all the heads-up coin *indices* computes all six parities at once: the ith bit of the result is precisely the parity of the coins whose index has a 1 in position i, which is exactly the parity of S_i. Therefore the expression above collapses to

$$E(a) = \bigoplus_{\substack{j=0 \\ a_j = 1}}^{63} j = s_1 \oplus s_2 \oplus \cdots \oplus s_k, \quad \text{where } \{s_1, \ldots, s_k\} = \{j : a_j = 1\}.$$

[5]This construction is actually the basis of a magic trick I loved as a child – you would show an audience six cards covered in numbers, ask them to think of a number between 1 and 63, and have them point to the cards containing their number. The cards they choose directly reveal the binary representation of the number, letting you reconstruct it instantly, and boom! People think you're from Hogwarts.

In other words, the prisoner simply XORs together the indices of every heads-up coin, and reads off the result. To find which coin to flip, note that $m \oplus i = t \iff i = m \oplus t$, which is always a valid index in $\{0, 1, 2, \ldots, 63\}$, so the strategy is always guaranteed to work.

What a satisfying solution! We began with a failed attempt at modular arithmetic, found the key insight in parity, and arrived at a clean algorithm expressed with a single XOR operation. Congratulations on making it this far, but plot twist! We are far from done...

4 The General Case

We have now established that an encoding strategy exists for $n = 2, 4$, and 64 coins. A natural question is whether this generalises to arbitrary n.

4.1 Exactly 1 coin flip

Our construction extends immediately to any $n = 2^k$: grouping tile indices by each of their k binary digit positions always yields k valid sets, and the same argument goes through unchanged.

One might hope that a sufficiently clever encoding could handle *any* n. After all, with n coins, the prisoner can flip any one of n coins – surely this provides enough information to convey n distinct states from any starting configuration?

Unfortunately, this is not the case, and we can prove it. Consider a *state diagram*: a graph where each node is a coin configuration, edges connect configurations differing by exactly one flip, and each node therefore has exactly n neighbours.

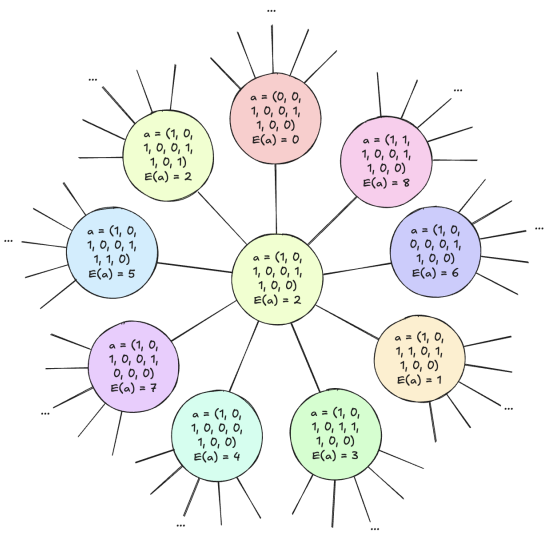

Figure 5: Example State Diagram for $n = 9$

Now suppose we have a valid encoding assigning each of the 2^n nodes exactly one of the n tile labels. For the strategy to work, every node must be adjacent to a node carrying each possible label, and since there are exactly n neighbours and n labels, each label must appear exactly once among them.

This gives the following double counting argument. Fix any label $\ell \in \{0, \ldots, n-1\}$, and let c_ℓ denote the number of nodes carrying label ℓ. Consider all edges in the graph with at least one endpoint carrying label ℓ, counting each endpoint separately. On one hand, every node has exactly one label-ℓ neighbour, contributing 2^n such endpoints in total. On the other hand, each of the c_ℓ nodes carrying label ℓ has degree n, contributing $c_\ell \cdot n$ endpoints. Hence $c_\ell \cdot n = 2^n$, so $c_\ell = 2^n/n$. For this to be a positive integer for every ℓ, we require $n \mid 2^n$, which holds iff n is a power of 2.

For all other values of n, no valid encoding can exist, and therefore no guaranteed strategy is possible[6]. I find that simultaneously so elegant and a little sad.

4.2 Up to 1 coin flip

Now suppose the prisoner may also do nothing. In our XOR construction, tile 0 was effectively a 'do nothing' move: since $a \oplus 0 = a \ \forall a$, flipping coin 0 leaves the encoding unchanged. Removing it and considering $n = 2^k - 1$ with tiles labelled $1, 2, \ldots, 2^k - 1$ leaves all sets S_i intact, since none of them contained 0 to begin with. The same encoding then applies directly: if the board already encodes the correct tile, the prisoner does nothing; otherwise they flip the appropriate coin as before.

Beyond $n = 2^k - 1$, however, the single extra option of doing nothing cannot extend the strategy further. To gain more flexibility, we need to allow more flips.

4.3 Up to 2 coin flips

With up to 2 coin flips, the prisoner can reach

$$\binom{n}{2} + \binom{n}{1} + \binom{n}{0} = \frac{n^2 + n + 2}{2}$$

distinct configurations from any starting state[7]. This means up to $\log_2\left(\frac{n^2+n+2}{2}\right) \approx 2\log_2(n)$ bits of information can be conveyed, more than the $\log_2(n)$ bits required. Intuitively, a guaranteed strategy should always exist. We now prove this using the same XOR-based encoding.

Consider any n with $2^k < n < 2^{k+1}$, so the tile set is $\{0, 1, \ldots, n-1\}$ and in particular contains 2^k. Suppose $E(a) = m$ and the target tile is t, so we seek indices $i, j \in \{0, \ldots, n-1\}$ such that $m \oplus i \oplus j = t$, or equivalently $i \oplus j = m \oplus t$.

[6] When researching this problem, I came across 3Blue1Brown's video [2], which uses a closely related argument. Rather than a general state diagram, it pictures each binary string as a vertex of an n-dimensional hypercube, which is a wonderful mental image.

[7] Corresponding to flipping 2 coins, 1 coin, or 0 coins respectively.

Case 1. $m \oplus t < 2^k$. Take $i = m \oplus t$ and $j = 0$. Then $i \oplus j = m \oplus t$ as required, and since $i < 2^k \leq n$, this is a valid choice.

Case 2. $m \oplus t \in [2^k, n)$. Take $i = (m \oplus t) \oplus 2^k$ and $j = 2^k$. Both $m \oplus t$ and 2^k lie in $[2^k, 2^{k+1})$ and therefore share a 1 in the kth bit position – their XOR clears this bit, giving $i < 2^k \leq n$. So i is again a valid tile index.

In both cases we can find valid i and j, so the two-flip strategy works for all n. Rather neat!

5 Conclusion

I hope, dear reader, that you have enjoyed this journey: from an apparently impossible puzzle, through failed modular arithmetic, to the elegant insight of parity, and finally the satisfying concision of bitwise XOR. This problem has a remarkable way of drawing together ideas from across mathematics, and I found genuine joy in exploring each step.

It is also worth noting that this puzzle connects directly to *Hamming codes* and error correction theory, which use the same parity-based construction to detect and correct errors in data transmission [3]. If you want to explore further, I would highly recommend the videos on this topic by 3Blue1Brown and Matt Parker, as I am sure you will come away with a fresh perspective or two.

And with that, you and your prisoner friend have finally escaped mathematical jail! The warden reluctantly frees your handcuffs and you step out into the warm golden sun. What mathematics will you explore next with your newfound freedom?

References

[1] Parker, M. [Stand-up Maths]. (2020, July 6). *The almost impossible chessboard puzzle* [Video]. Youtube. https://www.youtube.com/watch?v=as7Gkm7Y7h4

[2] Sanderson, G. [3Blue1Brown]. (2020, July 7). *The impossible chessboard puzzle* [Video]. Youtube. https://www.youtube.com/watch?v=wTJI_WuZSwE

[3] Hamming, R. W. (1950). *Error Detecting and Error Correcting Codes*. Bell System Technical Journal, 29(2), 147–160.