Spring 2016

Zero-Knowledge Proofs for Puzzles

Hunter Ripsom-Gardiner
ripsomgardih@merrimack.edu

Zero-Knowledge Proofs for Puzzles

An Honor's Thesis
Submitted in Partial Fulfillment of the Requirements
For the Degree of
Bachelor of Science with Honors
Computer Science

Department of Computer Science
Merrimack College
315 Turnpike St.
N. Andover, MA

Hunter Ripsom-Gardiner
Supervisor: Dr. Zachary Kissel

Chapter 1

Introduction and Background

1.1 Motivation

Why do we care about zero-knowledge proofs? The reason is simple: a zero-knowledge proof is a way of presenting a proof that only reveals the validity of the proof. For example, say Alice wants a password to access a computer. Bob claims that he has the password to that computer and wants Alice to pay him for that knowledge. If Alice pays Bob she is not guaranteed that the password he has is correct, in which case she may waste her money. If Bob gives Alice the password to prove that it is correct, he is not guaranteed Alice will pay him. Using a zero-knowledge proof Bob is able to prove to Alice he has the password without actually revealing what the password is to Alice. After completing the proof, Alice will be assured that Bob knows the password and will pay him so that he presents the password to her.

1.2 What is a Zero-Knowledge Proof?

A zero-knowledge proof is a type of interactive proof. Interactive Proofs involve two or more parties communicating with each other. Where a standard proof will list out a set of facts and draw a conclusion from them such as one would find in a text book, an interactive proof is closer to a student asking questions of a professor in order to understand that conclusion. Interactive proofs will always be between two parties: the prover and the verifier. The prover attempts to convince the verifier that the knowledge presented in the proof is true, while the verifier ensures that the information presented by the prover are in-fact true. Note that the prover has exponential time to perform calculations *before* the interaction, but only polynomial time during the interactions. The verifier only has polynomial time to perform calculations at all times.

Interactive proofs have two requirements:

1. Soundness: The verifier accepts a false statement as true only with *negligible* probability. A function is negligible if it is less than or equal to any inverse polynomial. This

will be explained in more detail later, but in all interactive proofs it is possible for a false statement to be proven true, but it must be possible through the proof to reduce that chance to almost zero.

2. Completeness: The verifier always accepts a true statement as true.

The zero-knowledge property states that the only information learned by the verifier are the truth (or falseness) of the statement and information the verifier could readily learn on his own. If an interactive proof has this third property, then it is called a zero-knowledge proof.

Let us examine a simple zero-knowledge proof to analyze these requirements.

Victor, the verifier, is colorblind and goes to a clothing store to buy one red sock and one green sock for Christmas. Unfortunately, due to his colorblindness Victor can't tell if two given socks are the same color or not, so he asks the store clerk Peggy, the prover, to help him. Peggy fetches one red sock and one green sock, but Victor still has no way of knowing if the socks are different. Victor has Peggy identify the red sock and places it in his right hand, with the green in his left hand. The act of Peggy giving the socks to Victor is known as "committing" in a zero-knowledge proof.

The commit is the point in the proof where the prover does some amount of work to "lock in" her answer to a question to be asked by the verifier. In most zero-knowledge proofs this is done by having the prover send to the verifier data encrypted using a chosen plain-text attack (CPA) secure encryption algorithm. The CPA secure encryption ensures that the same message will result in different cipher texts. A *non*-CPA secure algorithm, such as a block cipher was used used to encrypt a string of a's and b's, where each letter represents a block, then the verifier would be able to distinguish in the message "aaaabb" that more of one block type were encrypted than the other. If the zero-knowledge proof used numbers or a phrase repeatedly, then the verifier could potentially gain information he should not have from the interactions. The use of a CPA encryption algorithm ensures that regardless of what information is present before encryption, the verifier will be unable to detect patterns in the data after encryption. In this physical zero-knowledge proof, Peggy commits to her knowledge by letting Victor hold the socks, after that point Peggy has no way of changing the socks to another pair from the shelf or otherwise modifying the socks in question.

Victor has Peggy turn around and he flips a coin, based on the result he either swaps the two socks or keeps them in their current position. Victor then presents the two socks to Peggy again and has her identify the red sock. Assuming that the socks are different and that Peggy is not also color blind this should be easy for her to do. However, let us imagine that the socks were both red. Peggy will only have a 1-in-2 chance of identifying the sock Victor believes to be red. If she fails, Victor will know she lied and not buy the socks. To ensure that Peggy is not cheating, Victor repeats the process of flipping a coin and presenting the socks to Peggy many times. Again, if Peggy is not cheating it should be trivial for her to answer correctly every time. But if she was cheating, each round increases the chance that Peggy will be caught lying. After n rounds of questioning, the chance a lying Peggy will not be caught is 2^{-n} as each exchange is an independent event. As the number of rounds of questioning increase the chance of her lying approaches zero. So while it is *possible* for Peggy to be lying, the likelihood of it is negligible. Thus, when Victor is satisfied that the chance of Peggy lying is sufficiently small he will accept that she is telling the truth and buy the socks.

All zero-knowledge proofs have some amount of randomness in the set up of the interaction or in the questions asked during the interaction. This means that over several rounds of interaction, even if the prover is lucky and is not caught immediately when answering a question, the lying prover should be caught with high probability as the interactions increase. Thus we have soundness.

If the prover is telling the truth, such as the socks being different colors, there should be nothing in the interaction that could lead to the truth being interpreted as a lie. Thus we have completeness.

The proof must be constructed in such a way that the only information the verifier learns is information he could reasonably come up with on his own. This is the zero-knowledge property.

To make this a zero-knowledge proof Victor's interaction with Peggy must be computationally indistinguishable from a simulation of interacting with Peggy. To do this we use a simulator.

The simulator takes the place of the prover (hence it simulates the prover). Note that the use of a simulator is a proof technique to make sure a zero-knowledge proof is actually zero-knowledge, and the verifier would not actually interact with the simulator during a zero-knowledge proof itself.

There are several differences between the simulator and the prover: first, the simulator is allowed to only have limited knowledge of the solution (as opposed to knowing the full solution to the problem). Second, during the interaction, whenever the simulator is asked a question outside of its limited knowledge, the transcript of the interaction is rolled back and the verifier is forced to ask a new question instead, thus the question that was rolled back is not added to the transcript. The goal is that the transcript between the simulator and the verifier, and between the prover and the verifier should be computationally indistinguishable for the same questions. Computational indistinguishability means that given two transcripts, one between the prover and verifier, the other between the simulator and the verifier, there should be no way to determine which transcript belonged to which interaction in polynomial, in the size of the interaction, time. To make this happen, before every interaction the simulator decides what questions it has the answers to, and will answer those questions during that interaction. Then during the next interaction, the simulator decides again what question it can answer for that next interaction.

In the interaction above Victor would simulate Peggy and ask himself questions. Before each question the simulation of Peggy would determine what it knows (in this case which sock is red and which is green) then Victor would ask the simulation of Peggy if the socks have been switched. The simulated Peggy will be able to answer the question for that round because the simulator had that answer. For the next round the simulator again decides what it knows and the process repeats. Note that for this example Victor only asks one question and the simulator only knows one fact, so the process of the simulator deciding what it knows is straightforward. In other zero-knowledge proofs the simulator might know more information because Victor may ask more than one question.

If the transcript between Victor and Peggy was compared to a transcript between Victor and the simulated Peggy they would be computational indistinguishable. Victor gains no information from the commits, the questions asked in each transcript are the same, and the Victor's behavior is the same as well. Thus because Victor learns nothing from Peggy other than what he could learn from the simulator, the interaction meets the Zero-Knowledge Property.

1.3 Zero-Knowledge Proofs and NP-Complete Problems

By definition, any NP problem can be reduced to an NP-Complete Problem using a polynomial time reduction. Thus if there was a zero-knowledge proof for a single NP-Complete problem, π, there would be a way to spend polynomial time to reduce any NP problem to π and use its zero-knowledge proof protocol. Such a zero-knowledge proof exists, for vertex three colorability, which is NP-complete. Thus, all NP problems have a zero-knowledge proof [6].

Let us examine the interaction between Peggy and Victor for the vertex three colorability zero-knowledge proof. Peggy and Victor are presented with a graph $G = (V, E)$ which Peggy claims is three-colorable. This means that using only three colors it is possible to color G such that no two vertices that share an edge will have the same color. Victor does not believe her so they begin a zero-knowledge proof. Peggy solves the vertex coloring for the graph and randomly recolors her solution. For each of the the three colors she used on her original graph, Peggy substitutes a new color. So C_1 becomes C_1', C_2 becomes C_2', and C_3 becomes C_3'. This is to ensure that Victor cannot determine what the original solved three-colored graph looks like when she communicates with him. Peggy then encrypts each vertex's color and sends them to Victor. This is the *commitment.*

Victor selects an edge uniformly at random from E and requests that Peggy reveal the vertices on that edge. Peggy sends the keys to decrypt those two vertices, and *only* those two vertices. This is called *decommitment.*

Victor verifies that the two vertices are different colors. If they are not, he rejects, declaring Peggy a liar.

If Peggy was lying, she would only be able to ensure that some of the adjacent vertices are different colors. Thus the chance she is caught while lying is always at least $|E|^{-1}$ where $|E|$ is the number of edges in the graph. As the number of rounds increases the chance of Peggy being caught increases and after n rounds, the chance of being caught becomes $1 - |E|^{-n}$.

This process of recoloring, committing to an answer, and asking for an edge is repeated until Victor is satisfied that the chance Peggy is lying is negligible.

Soundness: As the number of rounds increase the likelihood that a lying Peggy is caught increases.

Completeness: If Peggy is telling the truth, then every time Victor asks for an edge the two vertices that are revealed will always be different colors.

Zero-Knowledge: Because Peggy recolors her graph each round, Victor has no way of identifying what the final proper coloring of the graph is for graph three-colorability, only that each individual pair of adjacent vertices are different. Formally, one can find a simulator to provide formal proof of this assertion.

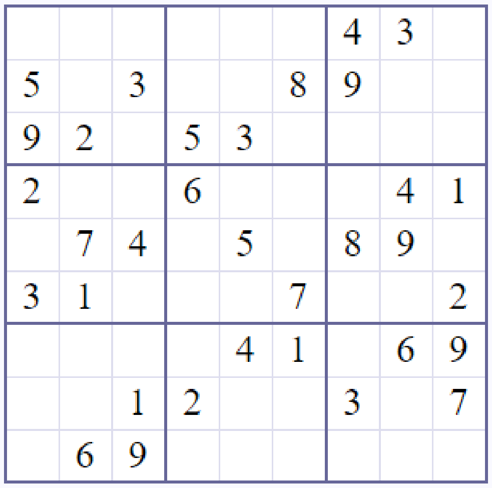
Thus this proof meets the requirements for a zero-knowledge proof, and any other NP problem could be reduced in polynomial time to vertex three colorability to use this zero-knowledge proof.

1.4 Zero-Knowledge Proof's for Puzzles

We now look at two zero-knowledge proofs for puzzles. These zero-knowledge proofs provided insight into how our zero-knowledge proofs for other puzzle games could be generated.

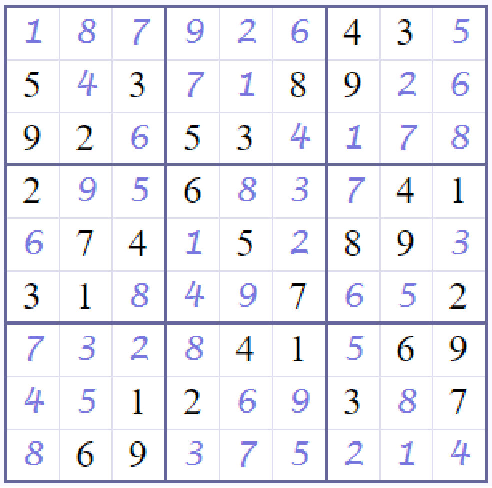
1.4.1 A Zero-Knowledge Proof for Sudoku

Sudoku is a puzzle game involving a 9×9 grid of squares, where each square can hold a number from 1 to 9. Some number of of the squares in the grid are already filled in with numbers. In order to solve the puzzle, one must fill in all the squares so that each row and column contains the numbers 1 through 9 without repetition. Furthermore, the grid has nine 3×3 sections, each of which must contain the numbers 1 through 9. Below in figure 1.1 we see an unfilled Sudoku board and in figure 1.2 we see that same board after it has been completed.



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

Figure 1.1: An Unsolved Sudoku Puzzle[1]



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

Figure 1.2: The Solved Sudoku Puzzle[1]

This zero-knowledge proof was discovered by Ronen Gradwohl et. al. [7] and Lance Fortnow from the blog "Computational Complexity" [4].

Peggy and Victor are presented with a Sudoku puzzle that Peggy claims to have solved. Peggy takes her puzzle and relabels each number of her puzzle then encrypts each individual square of the puzzle and sends those encryptions to Victor. Victor then may ask one of four types of questions:

1. To see a given row.
2. To see a given Column.
3. To see one of the 3×3 sections of the puzzle.
4. To see Peggy's relabeling of the board.

Peggy then sends the keys so that Victor can view the squares to answer the query. If the information matches what Victor expects (the numbers 1 through 9 for the first three questions, and a relabeling of the board for the last), he accepts for that round; otherwise he rejects. If Peggy is lying, the best she can do is to set up the board to correctly answer all but one of the questions. As the number of rounds increases the likelihood of Peggy being caught increases as well.

Soundness: The number of rounds of questioning is high enough that the chance Peggy is lying becomes negligible.

Completeness: If Peggy is telling the truth, the every time Victor asks for a row, column, 3×3, or the recoloring, Peggy will be able to properly present an answer.

Zero-Knowledge: Intuitively, if Victor asks for a row, column, or 3×3 region he will get no closer to solving the puzzle because when he receives that information, the numbers are already recolored, preventing him from gaining insight about the rest of the puzzle. If Victor asks for Peggy's renumbering, it is equivalent to him having a new Sudoku puzzle to solve, and thus gets him no closer to the final answer.

1.4.2 A Zero-Knowledge Proof for a Rubik's Cube

The Rubik's Cube is a toy consisting of 6 sides made up of 9 smaller squares on each face as shown in figure 1.3. The squares make three layers, perpendicular to each face so that the Rubik's Cube appears to be made of 27 smaller interconnected cubes. The layers can be rotated so the stickers on the faces of the small cubes can be rearranged. To begin the puzzle, a solved Rubik's cube (where each face is made of 9 identical stickers), is permuting it by rotating some number of layers in various directions. The solving of the puzzle is the process of returning the cube to its original solved configuration where each side of the cube is a single color. Note that a Rubik's Cube can be generalized to having faces consisting from anywhere from 2 to n layers built into it, we are only concerned with the $3 \times 3 \times 3$ cubes for this paper.

A zero-knowledge proof for Rubik's Cubes was discovered by Emmanuel Volte et. al.[10]. We survey their proof below.

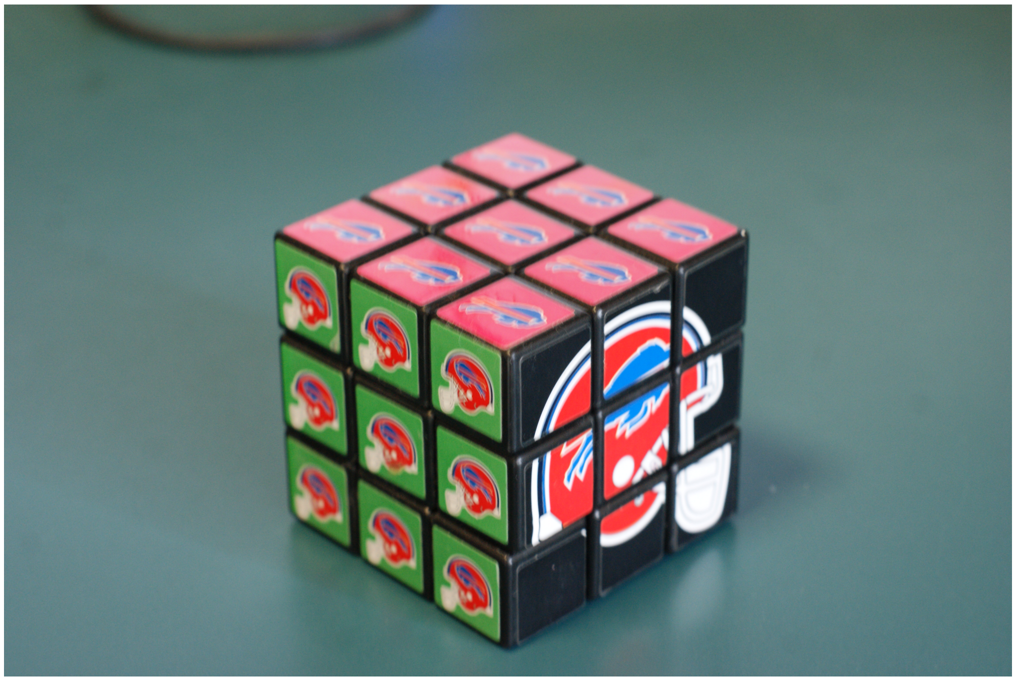


Figure 1.3: A Rubik's Cube

Peggy and Victor are presented with a Rubik's Cube that is in some unsolved configuration. Peggy claims that she has knowledge of what moves are required to return the Rubick's Cube to its solved configuration. For Rubik's Cubes it must be noted that there exist known moves or sets of moves in order to rearrange the smaller cubes into desirable configurations. For example, a cube is generally solved one layer at a time starting with the bottom, then the middle, and finally the top layer. What Peggy is really proving is her knowledge of these sets of moves and the general knowledge of how to solve the Rubick's Cube. It is possible for any person to solve a Rubik's Cube if they spend exponential time to use brute force to check every possible set of moves that could be made to reach the solved state of the cube. However, Peggy is only allowed polynomial time while interacting with Victor, so she must have knowledge of the puzzle and the ways to manipulate it in order to reach a desired configuration. Victor does not believe Peggy so they engage in a zero-knowledge proof.

Assume that there are two cubes that are identical in all ways, including the state of their faces. Victor takes both cubes and applies an identical set of transformations to both of them so that they are both in a new state. Victor gives both cubes to Peggy and she must apply her own transformations to them.

She must take one cube and solve it to completion from its new state. The other she must rearrange to undo Victor's transformations. She commits to both cubes by locking them in separate boxes and handing the boxes to Victor.

Victor flips a coin and based on the outcome asks for either the solved cube or the reverted cube. Peggy unlocks the requested cube and reveals it to Victor. If the cube is in the state Victor requested he accepts; otherwise he rejects.

This process is continued multiple times until Victor is satisfied that the chance Peggy is cheating is negligible. Assume for this explanation that there exists some easy method to return the cubes to their initial positions after each set of interactions so the interactions can be repeated.

Soundness: If Peggy were to cheat she could do no better than solving one of the Cubes if it was in a position to be brute forced with a small number of moves. This means that a cheating Peggy can do no better than a 1-in-2 odds of being caught for each interaction. Therefore, as the number of rounds of interaction increase the chance of catching a lying Peggy increases as well.

Completeness: If Peggy has knowledge of how to solve a Rubik's Cube she should have no trouble solving one cube to completion and reverting the other to its base state from before Victor permuted it.

Zero-Knowledge: Because Victor only sees the final form of the puzzle after Peggy manipulates it, he gains no knowledge on what moves Peggy makes in order to get to those states, thus he gains no knowledge on how to solve the configuration to reach the solved state.

Note that Victor and Peggy both assume the other abides by the rules of playing with a Rubik's Cube. In particular, neither manipulates the stickers, or dismantles the cube in order to re-build it in a desired configuration.

1.5 Overview of Thesis

This thesis will demonstrate zero-knowledge proofs for the puzzle games Kakuro and Rush-Hour. For Kakuro we will provide a classic zero-knowledge proof similar to the zero-knowledge proof for Sudoku. For Rush-Hour we will provide both a physical zero-knowledge proof and a classic zero-knowledge proof. These proofs share similarities with the zero-knowledge proof for the Rubik's Cube.

Chapter 2

Kakuro

Kakuro is a puzzle game that is a mathematical analog of a crossword. To solve the puzzle one must write the numbers 1 through 9 in each row or column such that the sum of the numbers equals the target number or clue shown before the row or column. Numbers can't be used twice.

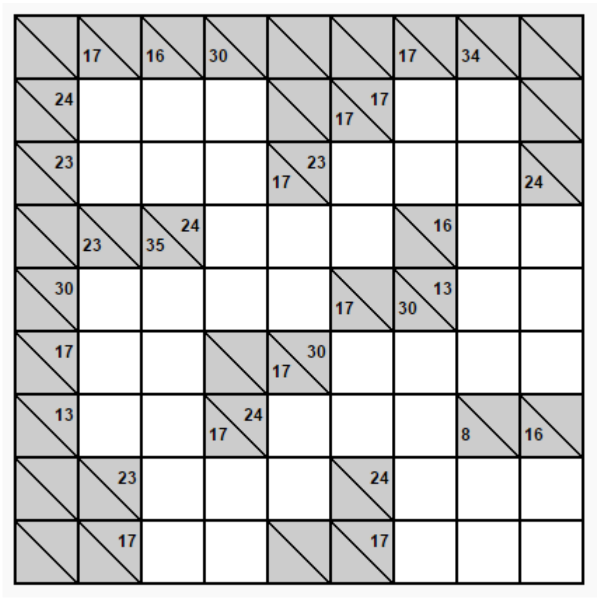


Figure 2.1: Unsolved Kakuro Puzzle

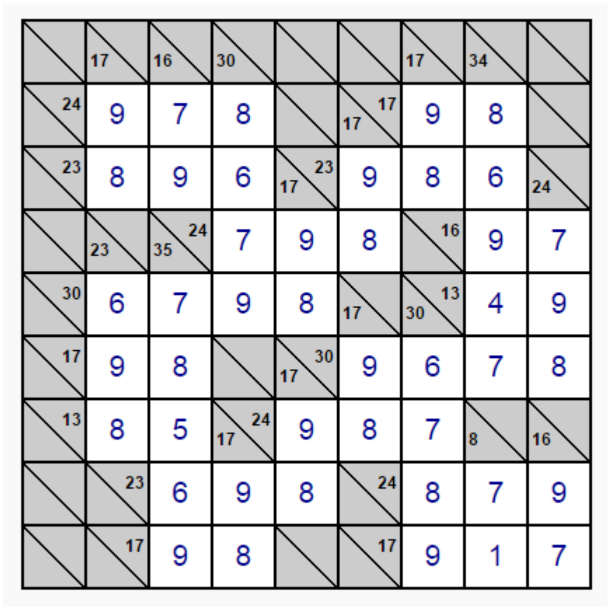


Figure 2.2: Solved Kakuro Puzzle

There are a finite number of solutions for each target number; for example in the puzzle shown above in figure 2.1, the 17 down in the top left corner (also circled in figure 2.3) must be made of two numbers, thus those numbers must be some ordering of 8 and 9. Similarly the 16 down to the right of 17 must be some ordering of 7 and 9 because 8 and 8 would be repeating a number, and thus illegal figure 2.4. Furthermore, in the second row from the top, 23 across requires 3 squares which must be 6, 8, and 9 in some order. From this we can see that the intersection of the 16 column and the 23 row must be 9 because it is the only number they share. The remainder of the puzzle is completed using a similar method of finding and eliminating solutions. The fully completed puzzle can be seen in figure 2.2.

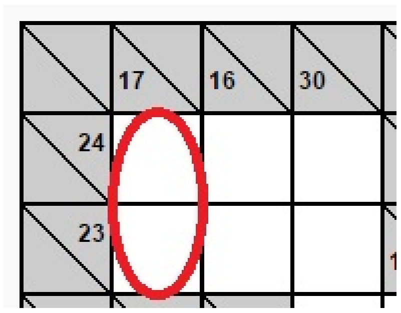


Figure 2.3: 17 over two squares

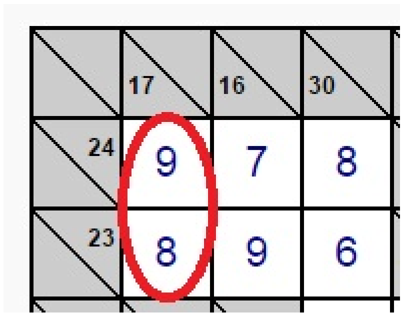


Figure 2.4: 17 Must be 8 and 9

Because there are a finite number of solutions to each target number, a table of these solutions can be generated. We will refer to this table as the *cheat-sheet*. The number of squares to be used changes the number of possible solutions. For example, 17 over two squares (which means the target number of 17 divided over two squares to be filled in) must be 8 and 9, but 17 over three squares can be any permutation of the entries in each tuple of: $(1, 7, 9)$, $(2, 6, 9)$, $(2, 7, 8)$, $(3, 5, 9)$, $(3, 6, 8)$, $(4, 5, 8)$, or $(4, 6, 7)$.

2.1 Kakuro's Zero-Knowledge Proof

Kakuro has been proved by Rupp et al. in [8] and T. Seta in [9] to be NP-complete. Therefore, it has a trivial zero-knowledge proof: reduce it to the vertex three colorability problem discussed in section 1.3 and utilize the appropriate zero-knowledge proof.

However, using a reduction for the interaction is complicated for most people to reason about, so instead we will present an non-reduction based interaction as shown below and in figure 2.5. To begin the interaction Peggy colors each square of the solved puzzle and the cheat-sheet by number, 1 becomes color C_1, 2 becomes color C_2, etcetera, while the target numbers are left untouched. The tuples on the cheat-sheet must be randomly permuted so that the no information is leaked to the verifier. If they are not randomly permuted then when information from the cheat sheet is revealed to Victor he can use the fact that the tuples are ordered to gain information about what colors in the puzzle correspond to which numbers, which would violate the zero-knowledge property of the proof. Peggy then encrypts each square of the puzzle and each tuple of possible answers on the cheat-sheet. She commits to the colorings by sending the encryptions to Victor using a Chosen-Plaintext-Attack (CPA) secure encryption algorithm, so that Victor won't be able to detect repeated occurrences of number or colors that Peggy uses in her commitments; after committing Peggy is unable to change her work.

Victor can then query any row or column (hence forth referred to collectively as zones), or he can query the cheat-sheet as a whole. The choice of querying a zone or the cheat-sheet

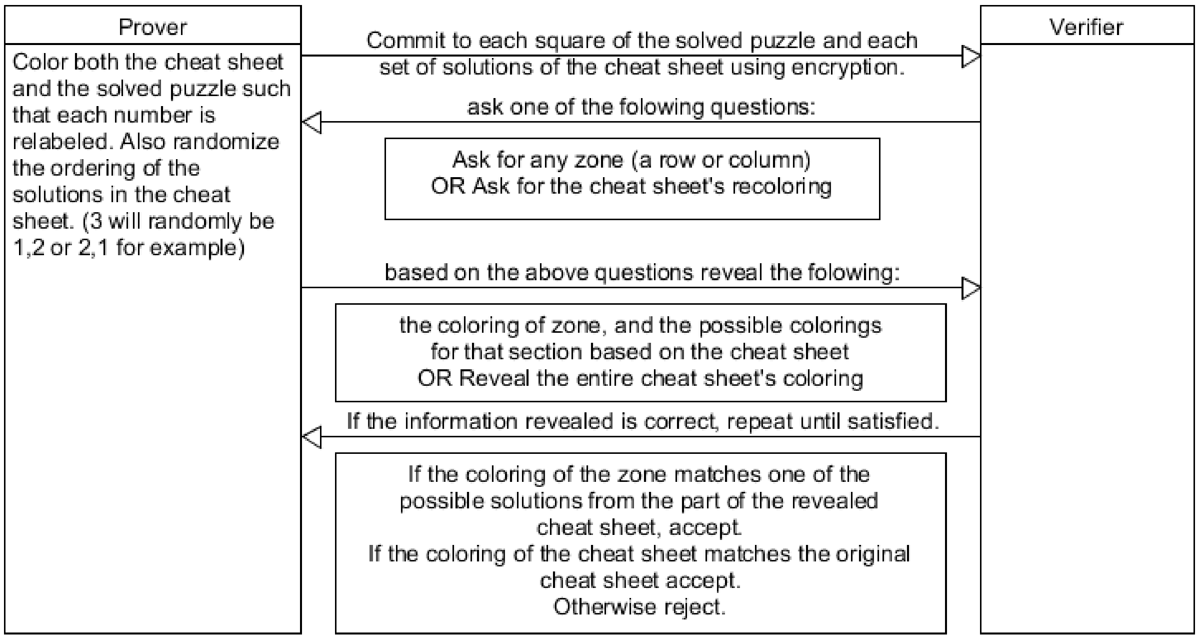

Prover
Color both the cheat sheet and the solved puzzle such that each number is relabeled. Also randomize the ordering of the solutions in the cheat sheet. (3 will randomly be 1,2 or 2,1 for example)
Commit to each square of the solved puzzle and each set of solutions of the cheat sheet using encryption.
ask one of the folowing questions:
Ask for any zone (a row or column) OR Ask for the cheat sheet's recoloring
based on the above questions reveal the folowing:
the coloring of zone, and the possible colorings for that section based on the cheat sheet OR Reveal the entire cheat sheet's coloring
If the information revealed is correct, repeat until satisfied.
If the coloring of the zone matches one of the possible solutions from the part of the revealed cheat sheet, accept. If the coloring of the cheat sheet matches the original cheat sheet accept. Otherwise reject.
Verifier

Figure 2.5: The interaction between prover and verifier for Kakuro

is chosen uniformly at random so that Peggy cannot predict what question Victor will ask. Furthermore, if a zone is queried, the specific zone that is queried is also chosen uniformly at random so that Peggy cannot predict Victor's behavior.

If Victor queries a zone, Peggy decommits the squares of that zone, as well as the section of the cheat-sheet referencing the target number for that zone by sending Victor the keys to the requested information. If the coloring of the zone matches one of the tuples for possible colorings on the cheat-sheet then Victor accepts for that round; otherwise he rejects.

If Victor queries the cheat-sheet as a whole, all the tuples from the cheat-sheet are decommitted; again by having Peggy send Victor the keys to the requested information. From this, if each color on Peggy's cheat-sheet can map to a single number on the original cheat-sheet, then Victor accepts for that round, otherwise he rejects.

When creating a zero-knowledge proof for Kakuro it is necessary to include the cheat-sheet in the proof. By including the cheat-sheet in Peggy's commitment we force her to substitute a single color for a given number in the puzzle. If she wanted to cheat the easiest way to do so is to recolor the board randomly, ensuring that no row or column has a repeated color. Without the cheat-sheet Victor would only ever see one row or column at a time and there would be no way for him to determine that Peggy was cheating. However, by requiring Peggy to also reveal part of the cheat sheet for the queried zone, it becomes possible for her to be caught. If Peggy recolored the board randomly, she would still have to fill in the tuples for her commitment to the cheat-sheet so that cheat sheet would match the board. So if Victor asked for the cheat-sheet as a whole or for one of the zones that Peggy did not rig to match the cheat-sheet, she would be caught.

When presented with the whole cheat-sheet (part of which is presented in figure 2.6) information will be leaked to Victor that the target numbers 3 and 4 share light blue in their tuples, thus the color light blue must correspond to 1 as shown in figure 2.7. This is intentional so that Victor can determine that Peggy is not cheating on the cheat-sheet. From that one can extract that pink must be 2 and light green must be 3. Victor has the ability to match the colors on the cheat-sheet to what the numbers should be and if things do not match up as they should then Victor catches Peggy in a lie. When only presented with part of the cheat sheet, for example if Victor asked for a zone with 7 as the target number over two squares, the colors Victor receives are permuted so that he will not be able to determine which numbers correspond to which colors.

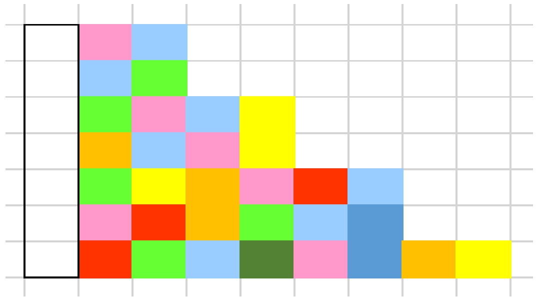


Figure 2.6: A filled partial cheat-sheet

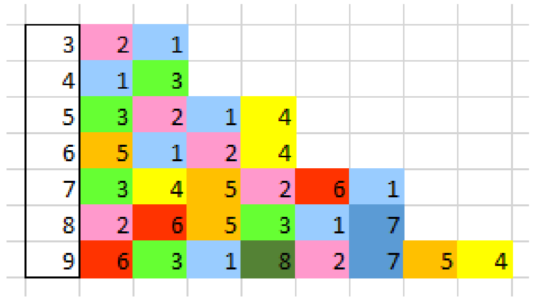


Figure 2.7: A filled partial cheat-sheet

2.2 Correctness of the Protocol

The protocol is repeated until Victor is satisfied that the likelihood of Peggy cheating is negligible. Assuming that Peggy is honest and has a solution to the puzzle in question, then Victor should always accept. Thus we have completeness.

However, if Peggy is lying and does not have a solution to the puzzle Victor should catch her in her lie. This is because it is impossible for Peggy to cheat on both the Kakuro board and on the cheat-sheet simultaneously. Thus, even if Peggy can successfully cheat on the all questions related to zones of the board (a target number and the associated permutations), she will still have a chance of being caught if Victor asks about the cheat-sheet. Since the cheat-sheet can be selected with uniform probability over q queries, Peggy can do no better than a $1 - 1/q$ chance of being accepted. Thus after k interactions the chance of being accepted is

$$(1 - \frac{1}{q})^k \tag{2.1}$$

As k increases, equation 2.1 will approach zero and thus the chance of Peggy being accepted approaches zero. Therefore, the chance of a lying Peggy being accepted is negligible as desired.

To find how many interactions we need, for a suitably small chance of accepting a false statement, say 0.001, we must solve $(1 - 1/q)^k = 0.001$ to determine a value for k. To do this we must solve the equation in terms of q. Thus we get:

$$k \log\left(1 - \frac{1}{q}\right) = \log 0.001$$

which solving for k becomes

$$k = \frac{\log 0.001}{\log\left(1 - \frac{1}{q}\right)}$$

Then we simply substitute in the number of possible questions for the given puzzle and solve for k, rounding up if needed. Therefore, we have soundness.

We claim the above interaction is a complete and full zero-knowledge proof. For this to be true, the interaction must meet the three requirements of a zero-knowledge proof: completeness, soundness, and the zero-knowledge property.

We have demonstrated the completeness and soundness above, we will now analyze the simulator and ensure that this protocol is indeed zero-knowledge.

There are two ways for the simulator to act between each interaction: the first way is that the simulator randomly colors the Kakuro board, then draws from the colored board to color as much of the cheat-sheet as possible. In this case, the simulator will roll back on questions for zones it does not have answers to, and for queries on the cheat-sheet itself. The second way the simulator can run is that it colors the tuples on the cheat-sheet, then randomly draws from those colorings to fill in the Kakuro board's horizontal or vertical zones (but not both, because that would require solving the whole puzzle). In this situation the simulator rolls back whenever it is asked about zones it does not know the answer to. In both situations the transcripts will be identical to the interaction between Peggy and Victor for the same questions. Because the commitments are encrypted using a CPA secure encryption algorithm, Victor, or any other outsider viewing the transcripts will be unable to detect if numbers or colors referred to in the puzzle are being repeated. Furthermore, because the questions asked in each transcript are the same, and the verifier's behavior is the same as well, we conclude that the transcripts must be computationally indistinguishable. Thus because the verifier learns nothing from the prover other than what he could learn from the simulator, the interaction is zero-knowledge.

Chapter 3

Rush-Hour

3.1 The Rules of Rush-Hour

Rush-Hour is a puzzle where many car shaped blocks are arranged on a finite grid. Each car is 2 or more blocks long and 1 block wide. Each car can move forward or back along its long axis as long as no other cars are in its way, and as long as it does not try to exit the bounds of the grid. The goal of the game is to manipulate the cars such that a designated car (referred to as the red car or goal car from now on) can be moved through an exit in the wall of the grid. When the red car exits, the puzzle is considered solved. Normally the cars can be picked up from the board in order to set up a given puzzle, but for this puzzle we assume that the board has the cars affixed to the surface so that they can still slide, but that none of them can leave the plane.

Based on the work of Flake et al. [3] we know that Rush-Hour is P-Space complete, which means it can be solved in an exponential amount of time with a polynomial amount of space. We know that a proof exists for Rush-Hour because an interactive proof exists for Rush-Hour [2]. As with Kakuro, Rush-Hour can be reduced to graph three-colorability, then the known zero-knowledge proof for graph three-colorability can be used for Rush-Hour.

3.2 Physical Zero-Knowledge Proof for Rush-Hour

The zero-knowledge proof for Rush-Hour is as follows: Peggy and Victor are given the same Rush-Hour puzzle with the pieces being fixed to the board so that they cannot be lifted or removed. Peggy claims that she knows the solution to the puzzle. For the sake of this proof there are two copies of the puzzle boards to be used. Victor takes the puzzle and permutes it by moving the pieces in a random fashion within the rules of Rush-Hour. Victor performs the same moves for both puzzles so that they end up in the same permuted configuration.

Peggy is given both identical permuted puzzles. She must solve one of the puzzles and return the other to its initial pre-permuted state. Peggy then locks each puzzle in its own box; this is the commitment. Then Victor flips a coin and asks one of two questions based on the result either,

1. to return the puzzle to its original unsolved form, or
2. to solve the puzzle.

The requested puzzle is released from the box and shown to Victor. If the puzzle shown matches the puzzle requested Victor accepts for that round, otherwise he rejects.

Let us make a few assumptions about the lying Peggy to better understand how often she will be caught. If Peggy does not understand Rush-Hour fully then she will only have a polynomial amount of time in order to make random moves in an attempt to commit to one of the desired boards. Assuming that the lying Peggy is not exceptionally lucky and able to make every move right for both boards, she must spend all her time on one of the two boards in an attempt to solve it. Alternatively, it may be possible for her to be able to see that the required moves are simple, such as Victor's permutation being only a single move. In either case the best a lying Peggy can do (ignoring nigh-impossible luck on her part) is to solve one of the two boards she must commit to. Thus she can do no better than a 1 in 2 chance of not being caught. As the number of rounds of interactions increase, the chance of not being caught decreases because each interaction is an independent event. After n rounds of interaction the chance of not being caught becomes 2^{-n} which can be made as close to zero as Victor desires, which means that after many interactions the chance that Peggy is lying and has not been caught is negligible.

3.3 A Classic Zero-Knowledge Proof for Rush-Hour

We can encode Rush-Hour using matrices and basic linear algebra. For example we can take the diagram of a Rush-Hour board shown in figure 3.1, and convert it to a matrix as shown below

$$\begin{pmatrix} 5 & 5 & 7 & 9 & 0 & 0 & 1 \\ 0 & 0 & 7 & 9 & 0 & 0 & 1 \\ 3 & 3 & 0 & 9 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 1 \\ 0 & 13 & 15 & 15 & 0 & 11 & 1 \\ 0 & 13 & 0 & 17 & 17 & 17 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The cars are all represented by odd numbers, the walls of the board are represented by 1's to ensure that the board is square to enable matrix multiplication. Matrix multiplication requires the same number of columns in the left matrix as rows in right matrix. This is easiest to ensure if we make both matrices square. The goal square for the red car (car 3) is the zero in the rightmost column. This is a representation of the board as a whole, but it is made by matrix addition on many matrices, one for each car and one for the board itself. This means that the matrix for the red car would be car three by itself. The cars must be odd numbers so that when the matrices are added together, an even number will occur if the cars overlap or are intersecting the walls, which will mean that either Peggy or Victor

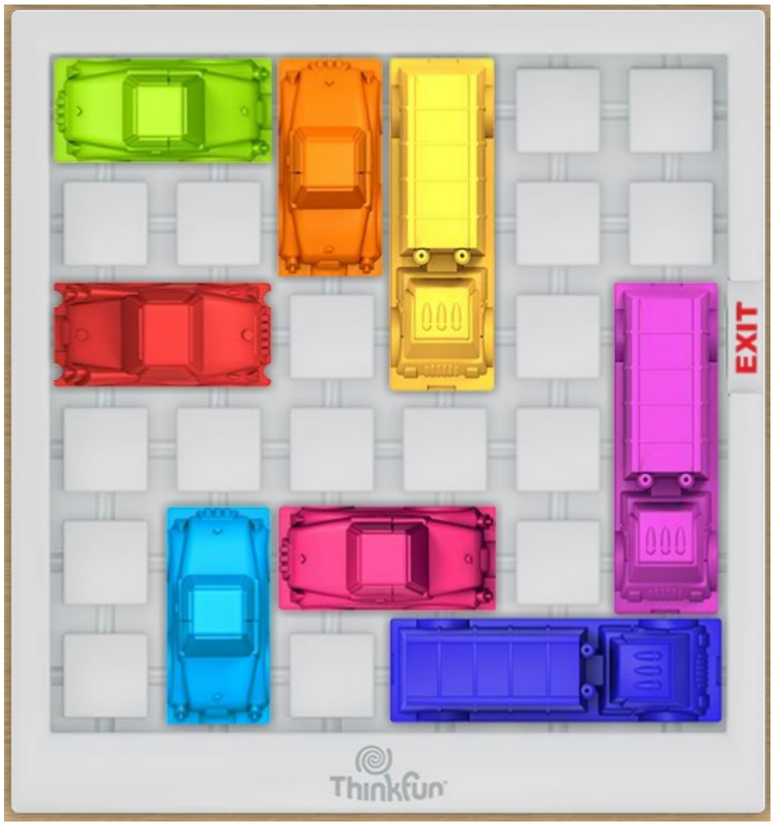


Figure 3.1: An unsolved board of Rush-Hour [5]

cheated while manipulating the cars in the puzzle. Cars are moved using transformation matrices. For example equation 3.1 shows the transformation matrix needed to move car 3 one space to the right:

$$\begin{pmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 3&3&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix} \times \begin{pmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix} \tag{3.1}$$

The row the 1's appear in is not important, so long as they appear in the columns that car three needs to be moved to and are positioned such that the 1's can absorb the car's value according to the rules of matrix multiplication. If car three was a vertical moving car then the transformation matrix would be right multiplied by the state matrix instead, and the ones would appear in whichever row the car needed to move through.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a*e+b*g & a*f+b*h \\ c*e+d*g & c*f+d*h \end{pmatrix} \tag{3.2}$$

As we can see in equation 3.2, the result of the matrix multiplication is the result of the components of the input matrices being multiplied and then added. Going row by row for the first matrix and column by column for the second matrix. Say we want o move a car horizontally. We must ensure that the squares that the car ends up in have the value for that car. This is very simple because we only need to ensure that a single 1 in the transformation matrix is positioned such that it will multiply one of the values labeling the car itself. To move car three shown here one space to the right we must ensure that the middle value of the second row on the result matrix is a three. To do that any of the squares marked with a T in the transformation matrix would be able to hold a 1.

$$\begin{pmatrix} 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 3&3&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{pmatrix} \times \begin{pmatrix} 0&0&T&0&0&0&0 \\ 0&0&T&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{pmatrix} \tag{3.3}$$

If we look at the original puzzle above we see that car 3 will intersect with car 9 after two shifts to the right. This results in an even number at the point of intersection when car 3 and car 9's matrices are added together. When the prover and verifier interact we make the assumption that a car cannot make a move that results in an even number appearing on the board (note that zero is neither odd nor even). We must make that assumption because if such a move was allowed there would be no way for Victor to detect that an illegal move occurred during the proof. Victor gets to see the end result of the puzzle, and the path a single car takes, but he doesn't have the ability to see if that path intersects with other cars as it is traveling. However, Peggy and Victor are both required to make moves such that a car moves only a single square at a time and Victor can determine if Peggy cheats by breaking this rule. Peggy is required to commit the list of moves she makes on each car to Victor. During his interaction with Peggy, Victor is able to ask for one of the transcripts for one of the cars. Victor looks at all the moves that car made and if the transcript shows that the car moves farther than it should be allowed then Victor will reject.

3.4 Digital Rush-Hour's Zero-Knowledge Proof

The interaction between Peggy and Victor are similar to how they interacted in real life with a few minor differences: The board to be solved is presented to Peggy and Victor as normal. Peggy claims she can solve it so Victor permutes the digital board per the rules of Rush-Hour and sends the permutation to Peggy. Peggy takes two copies of the permuted board and must return one to the initial state and solve the other from its permuted state.

Peggy can make a polynomial number of moves on each board of the two boards where a move consists of the following algorithm:

1. Split the board into its multiple car matrices. Because there is no overlap of cars at the beginning of this process, one can simply make a number of matrices equal to the number of cars and the board and fill each with zeros, and numbers in the positions of the corresponding car from the base matrix.

2. Apply a transformation matrix to one of those cars to move it one space in a given direction.

3. Record which car had what transformation applied to it into a move list array.

4. Recombine the individual matrices into a single board by adding all the individual matrices together. This is the point where the addition check is carried out to make sure cars do not overlap.

5. Repeat until solved.

In step four we are adding the assumption that no cars will ever collide or overlap because of the addition check. After all the moves have been made Peggy takes her list of transformation matrices, sorted by car and then by order and encrypts them along with the board and sends them to Victor. One board and set of lists for reaching the initial state are encrypted and sent. One other board and set of lists for reaching the solved state are encrypted and sent as well. As before Victor flips a coin and based on the result asks for either:

1. the initial state puzzle, or

2. the solved puzzle.

Peggy decomits to the data for the query by sending the keys for the requested data. If the puzzle is in the desired form Victor moves on to another question about the list of moves sent. Victor flips a number of coins to choose a car uniformly at random and asks to see the array of moves made by the prover for that car. Peggy sends the keys to the data for the moves for the requested car. If Peggy has not cheated the list of transformation matrices should show a set of 1's moving the car gradually one space at a time across the board. If they do not, Victor knows that Peggy has cheated and moved a car through another car. If the board presented matches the form Victor requested and if the car moved without hoping over spaces Victor accepts for that round; otherwise he rejects.

Note that when Victor permutes the board, the resulting permutation must be different each time. Otherwise Victor could make the same set of moves to permute the board, see the transformations for all the cars, and then use depth-first search or a similar method to use the transformations to find the solution to the puzzle. We will assume that if Victor is not trying to maintain the zero-knowledge property of the proof, then the proof will not proceed.

3.5 A Proof of Correctness

If Peggy understands the puzzle presented and the ways of manipulating a Rush-Hour puzzle in general, she should have no trouble "solving" the permuted puzzle in both directions. However, if Peggy is lying she will be caught. This is because a cheating Peggy must both solve the permuted puzzle, and return the permuted puzzle to its initial state. As discussed for the classic zero-knowledge proof for Rush-Hour in the best case, the lying Peggy could get one of the two boards right some of the time, but not both. Thus, the the chance the lying Peggy will be not be caught is again 1 in 2. And as before, as the number of rounds increases the chance of the lying Peggy not being caught is 2^{-n} where n is the number of rounds of interaction. This approaches zero as n increases, which makes the likelihood of the lying Peggy not being caught negligible as desired.

To determine how many rounds of interaction are needed for a given desired confidence, say 0.001 chance that peggy is lying, we simply use the equation $2^{-n} = 0.001$ and solve for n to determine the number of rounds required. This is quite simple as the only variable is n, which we can easily isolate:

$$-n \lg(2) = \lg 0.001$$

Simplifying this we get

$$n = -\lg 0.001 \approx 10$$

We of course round up the number of rounds of interaction required, so to have a 0.001 chance that Peggy is lying, we only require Peggy to have completed ten rounds of the interaction successfully. If we required more confidence (a smaller chance that Peggy is lying) we would of course increase the number of rounds as determined by the above equation.

Chapter 4

Conclusion

Zero-knowledge proofs are a subset of interactive proofs. What makes zero-knowledge proofs special is that they allow Peggy to prove to Victor that she knows some amount of information without revealing the content of what she knows. With this thesis it is now possible for Peggy to convince Victor that she knows how to solve either a game of Kakuro or Rush-Hour, without revealing to Victor the solution as a whole.

Further study can be done to see if there is an alternate way to perform a zero-knowledge proof for Rush-Hour that requires weaker assumptions, thereby increasing the strength of the proof. However, it appears that the assumptions for both the physical and mathematical form of the proof must effectively reduce down to "both Peggy and Victor will obey the rules of Rush-Hour." It is an open problem to ensure that both Peggy and Victor can tell if the other breaks the rules of the game.

With the help of my professor, Zach Kissel, two novel zero-knowledge proofs have been supplied for puzzles. Thus expanding the current literature in this area.

Bibliography

[1] Web sudoku. 2015.

[2] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in CryptologyCrypto88*, pages 37–56. Springer, 1988.

[3] Gary William Flake and Eric B Baum. Rush hour is pspace-complete, or why you should generously tip parking lot attendants. *Theoretical Computer Science*, 270(1):895–911, 2002.

[4] Lance Fortnow. Zero-knowledge sudoku. 2015.

[5] Think Fun. Rush-hour blitz. 2015.

[6] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[7] Ronen Gradwohl, Moni Naor, Benny Pinkas, and Guy N Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. *Theory of Computing Systems*, 44(2):245–268, 2009.

[8] Oliver Ruepp and Markus Holzer. The computational complexity of the kakuro puzzle, revisited. In *Fun with Algorithms*, pages 319–330. Springer, 2010.

[9] TAKAHIRO Seta. The complexity of cross sum. *IPSJ SIG Notes, AL-84*, pages 51–58, 2002.

[10] Emmanuel Volte, Jacques Patarin, and Valérie Nachef. Zero knowledge with rubik's cubes. *IACR Cryptology ePrint Archive*, 2012:174, 2012.