Zero Knowledge Proofs

Authors: Manish Goregaokar, Pradyot Prakash,
Ashish Anand, Soham Ganatra

April 5, 2016

What is it?

A zero-knowledge proof is an algorithm allowing a party to prove to another party that they have a solution to a given problem, without divulging any details about the problem. It requires no trust between the two parties.

Example scenario

- Alice has a hard problem she wishes to solve, say, a graph 3-coloring or a sudoku problem
- Bob has a solution. He wishes to prove to Alice that he has this solution.
- However, he does not want Alice to know the solution immediately
- He can use a zero-knowledge proof!

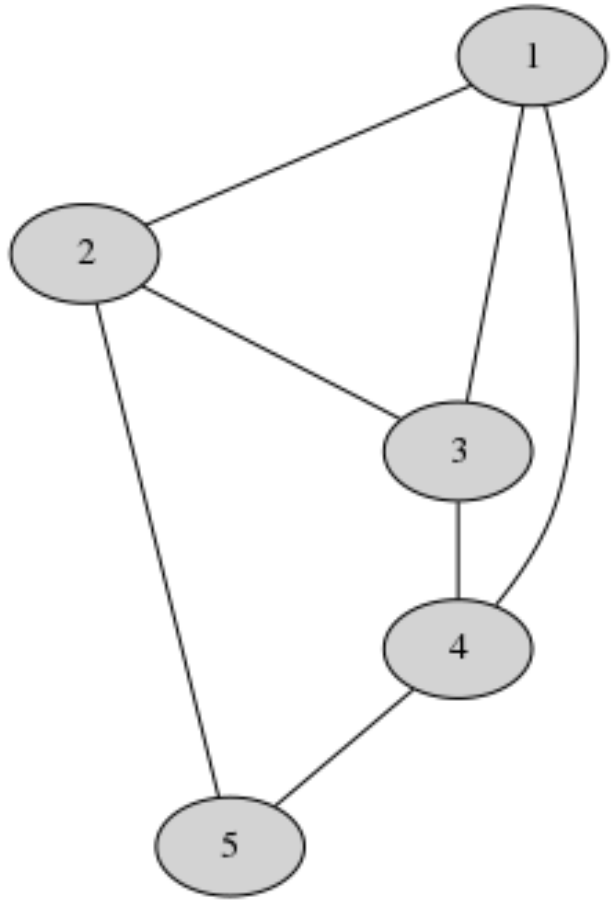
Coloring a graph

Coloring a graph

Three-coloring a graph (so that no two adjacent nodes have the same color) is a hard problem.

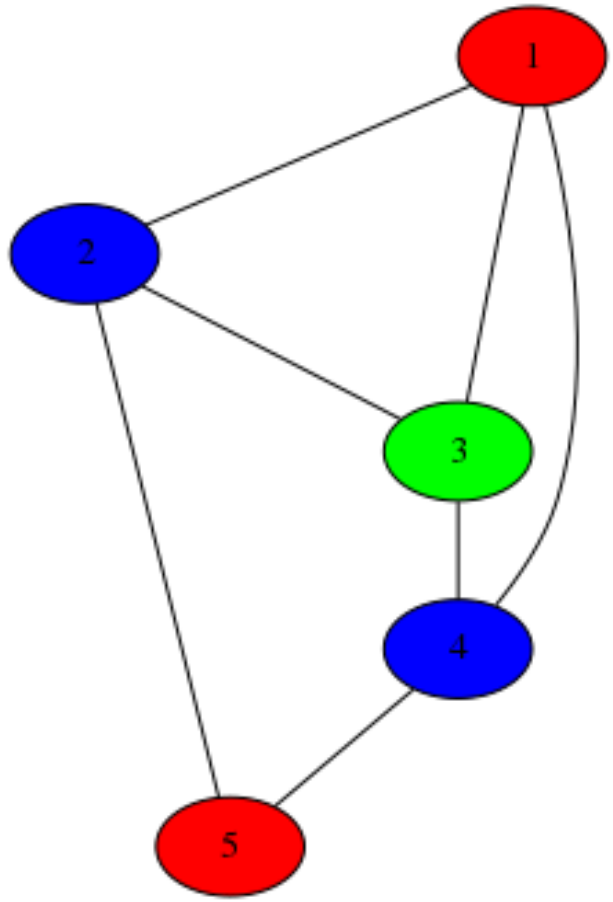
Coloring a graph

Alice has such a graph



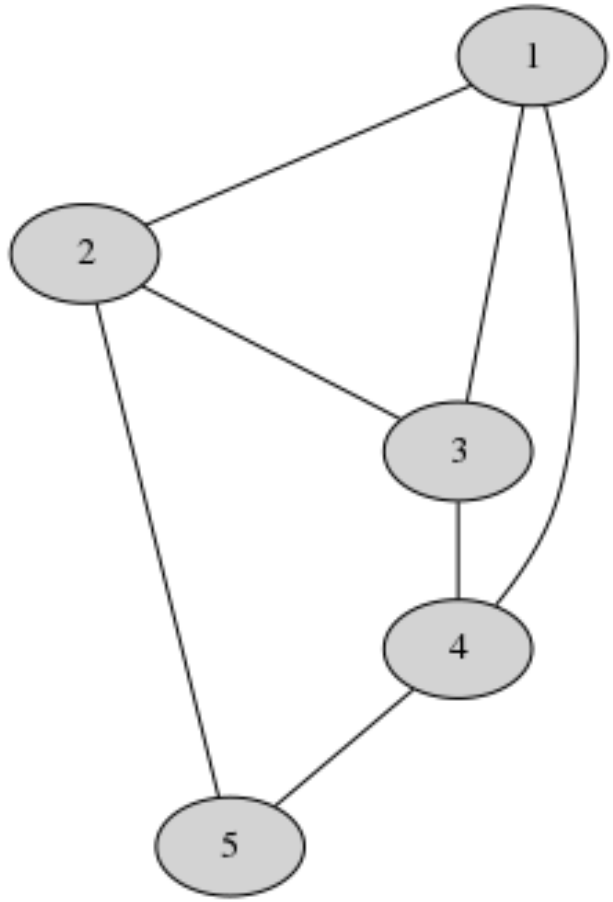
Coloring a graph

Bob has a solution.



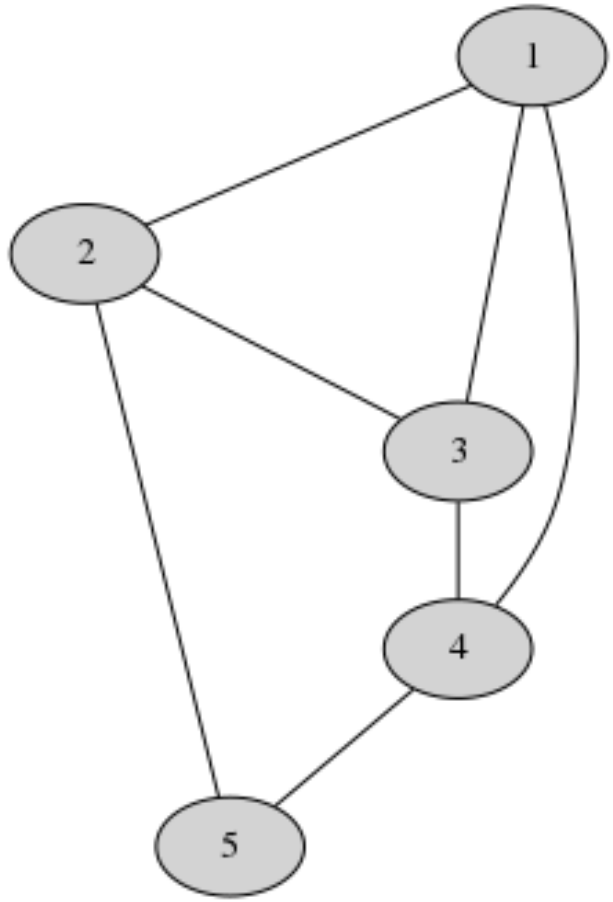
Coloring a graph

Alice draws the graph on a piece of paper and gives it to Bob, who goes into another room.



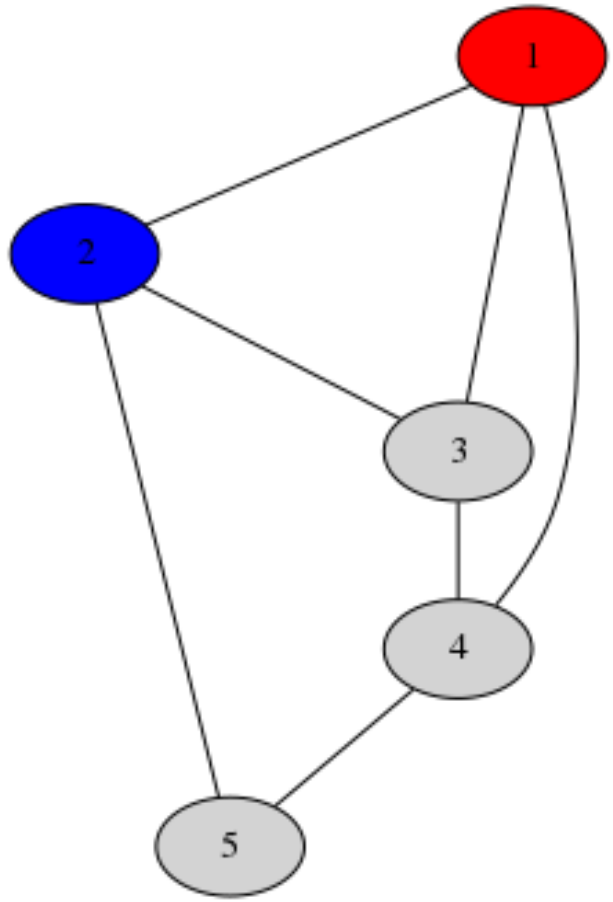
Coloring a graph

He colors it in, and then covers each node. He comes out of the room.



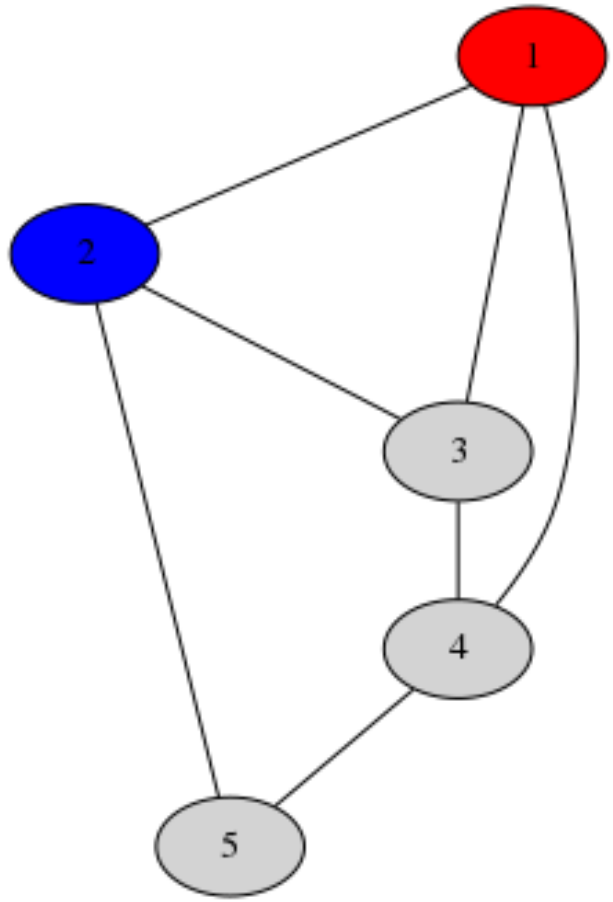
Coloring a graph

Alice now requests two adjacent nodes be uncovered.



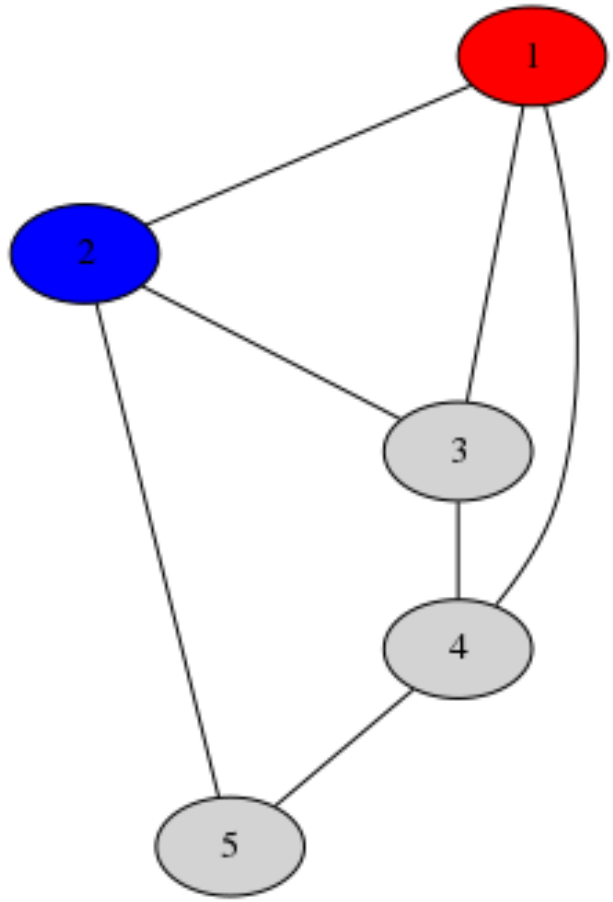
Coloring a graph

This is done in sight of Alice, so there can be no tampering of the colors.



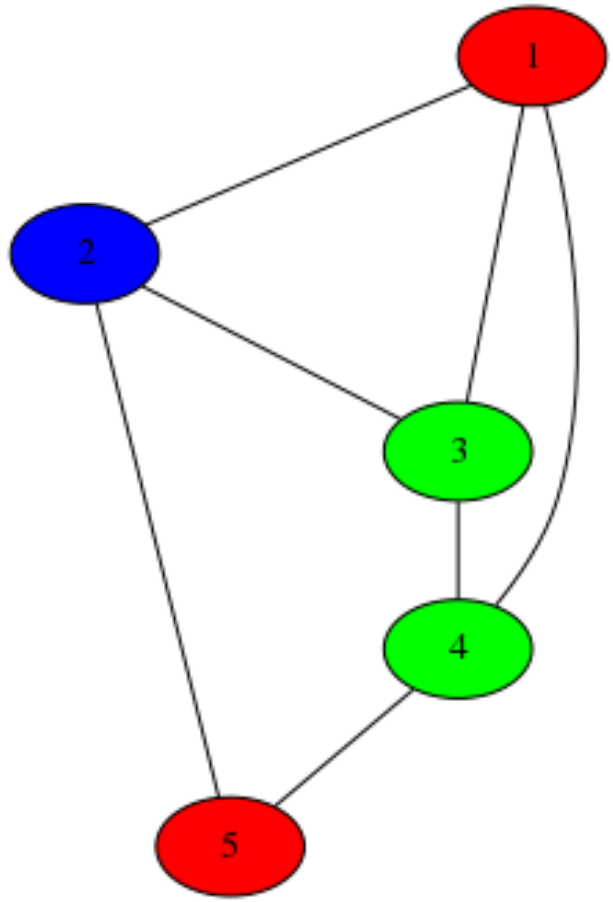
Coloring a graph

Alice can easily verify that the two adjacent nodes have a different color.



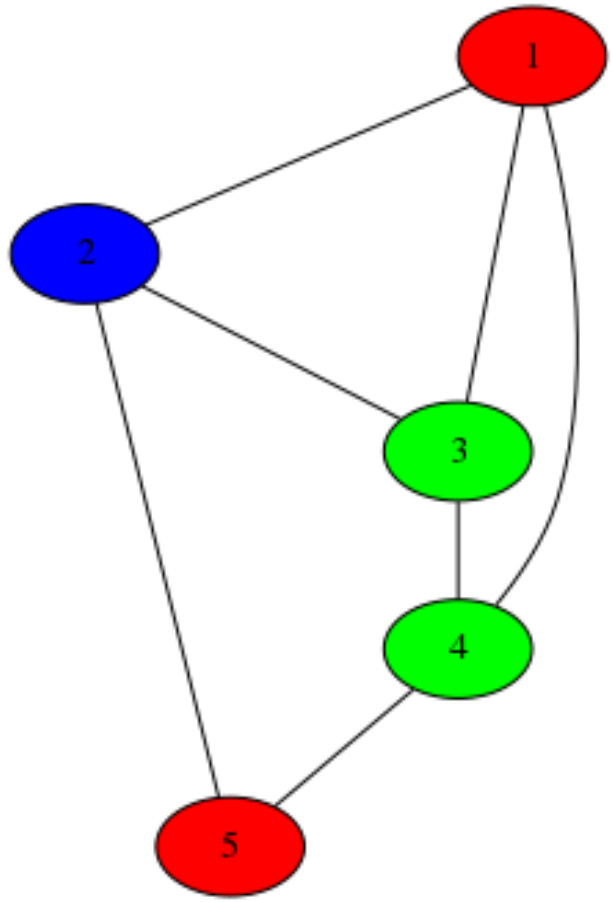
Coloring a graph

There's still a chance Bob's solution is wrong, from Alice's point of view



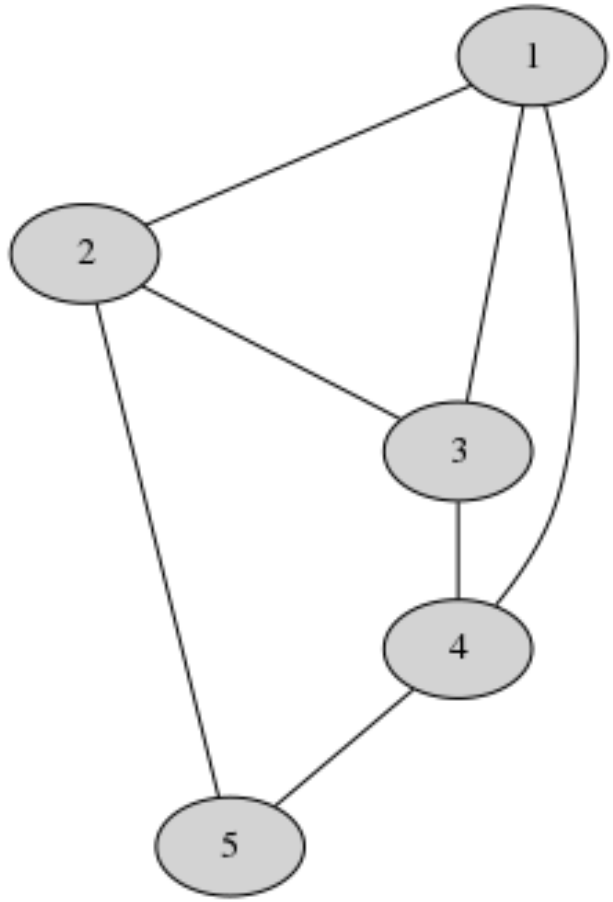
Coloring a graph

For the given wrong solution, Alice would have to choose the following nodes to detect the deception



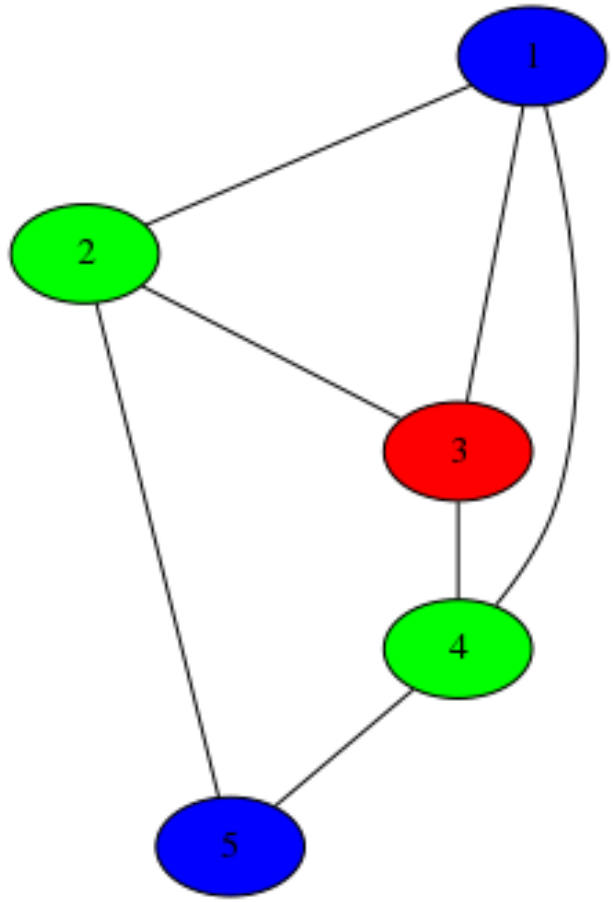
Coloring a graph

So, we try again



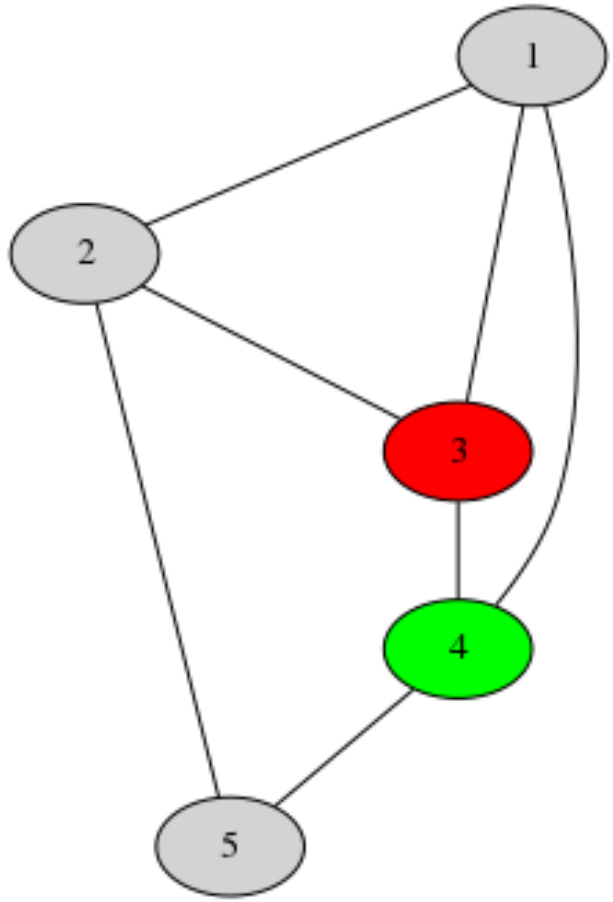
Coloring a graph

Bob first permutes his coloring,



Coloring a graph

Bob first permutes his coloring,
and Alice requests two adjacent nodes to be uncovered



Coloring a graph

Each round, Alice has a chance of revealing a deception.

Coloring a graph

As the number of rounds increases, Alice can become sure that there isn't any and that Bob has a correct solution

Coloring a graph

At the same time, each round, Alice only sees the information "these two nodes have different colors, as they should". No additional information about the coloring is leaked.

Issues

- This requires physical presence
- This requires physical trust (i.e. that Alice will not simply snatch the covered graph and uncover it herself)
- This is not scalable

Doing this the cryptographic way

We use hashes and nonces to make *commitments*. For each node, Bob chooses a random *nonce*, and hashes the concatenation of the nonce with the color. He shares these hashes with Alice. When he has to reveal the colors for two nodes, he simply reveals the nonces and colors for those nodes. Alice can verify that they match the hash. It is hard for Bob to fake values that produce a hash, so this has the same effect as covering the colors on a piece of paper – it can't be tampered with.

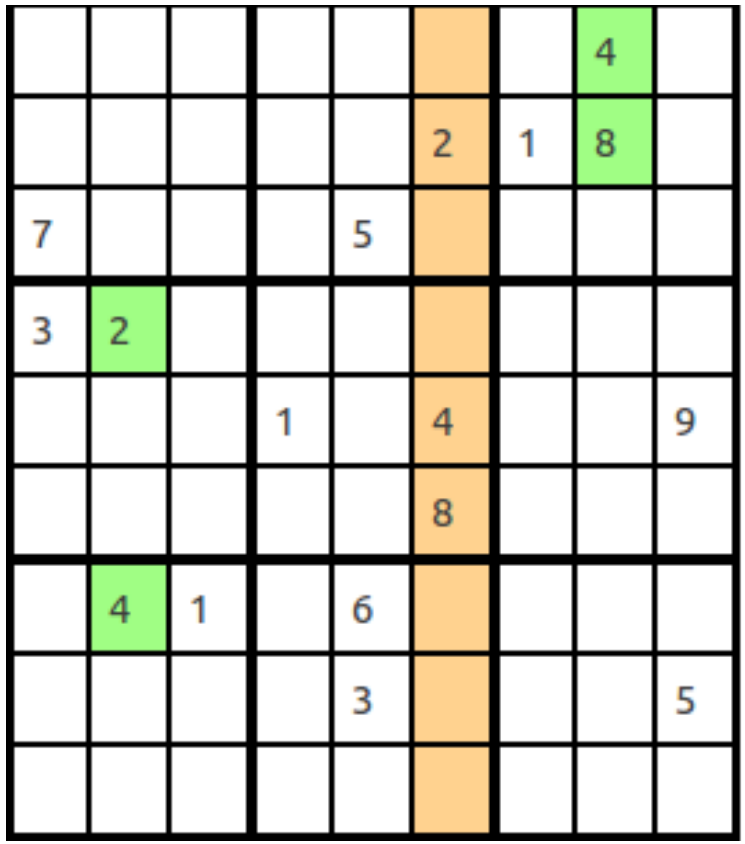
Solving sudoku

Solving sudoku

The algorithm is similar. Bob permutes the solution, and makes commitments to each individual cell. Alice asks him to reveal a single row, column, or 3x3 sub-square. If the row/column/subsquare contains pre-set values (from the problem statement), Bob must additionally reveal all other spots on the grid where these values are pre-set.

Solving sudoku

The algorithm is similar. Bob permutes the solution, and makes commitments to each individual cell. Alice asks him to reveal a single row, column, or 3x3 sub-square. If the row/column/subsquare contains pre-set values (from the problem statement), Bob must additionally reveal all other spots on the grid where these values are pre-set.



Solving sudoku

Alice then verifies if the row/column/subsquare has all 9 numbers in it, and if the pre-set values match up.

Demo

Zero Knowledge Contingent Payment

There is a general algorithm that lets one do a zero-knowledge proof for almost any problem. We've analysed this in depth in http://manishearth.github.io/blog/2016/03/05/exploring-zero-knowledge-proofs/

Zero Knowledge Contingent Payment

Using this algorithm, one can actually use zero knowledge proofs with bitcoin for securely swapping information for money. Bob can *sell* his solution to Alice without any trust required – Alice will not lose money if the solution is wrong, and Alice cannot see the solution unless Bob gets the money.

Zero Knowledge Contingent Payment

To do this, Bob first hashes his solution S to get hash H and shares this hash. A program is written which can verify the following *compound* statement:

- S is a solution to Alice's problem
- H is the hash of S

Using the general zero-knowledge proof algorithm, a zero-knowledge proof is executed with this program, and Alice is convinced that Bob has such a solution.

Zero Knowledge Contingent Payment

Now, Bob simply needs to securely swap the pre-image of H (that is, S) with Alice. The Bitcoin protocol can do this for us.

Zero Knowledge Contingent Payment

Each Bitcoin transaction has a *script*, which determines who can use a transaction.

Zero Knowledge Contingent Payment

Each Bitcoin transaction has a *script*, which determines who can use a transaction.Usually the script is “whoever can sign this transaction with the private key corresponding to the receiver’s address” .

Zero Knowledge Contingent Payment

Instead, we make the script two-fold, as "whoever can sign this transaction with the private key corresponding to the receiver's address, *and* has the pre-image of H".

The moment Bob attempts to take the money by providing the pre-image of H, this pre-image will be visible to the Bitcoin network and thus Alice. Since hashes are hard to fake, this will be the solution.

Further reading

- http://manishearth.github.io/blog/2016/03/05/exploring-zero-knowledge-proofs/
- https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/
- https://people.xiph.org/~greg/simple_verifyable_execution.txt
- https://github.com/scipr-lab/libsnark

Thank you!