bitcoin++
Istanbul

@nepetonium
github.com/nepet

Unveiling the Magic of Zero-Knowledge Proofs:

~~From Waldo to Bitcoin's Future~~

An introduction to the fundamentals of interactive zero-knowledge proofs

ZKPs: From Waldo to 3-Coloring

Today’s Journey:

What Are Zero-Knowledge Proofs?

- Building intuition with simple examples
- Interactive Proofs
- The three properties of ZKPs
- Deep dive: 3-Coloring Problem

The Fundamental Question

Can you prove you know something without revealing what you know?

Real-world analogies:

- Proving you're over 21 without showing your exact birthdate
- Proving you have sufficient funds without revealing your balance
- Proving you know a password without typing it

This seems paradoxical, but it's possible!

Where’s Waldo?
The Classic Example

The Scenario:

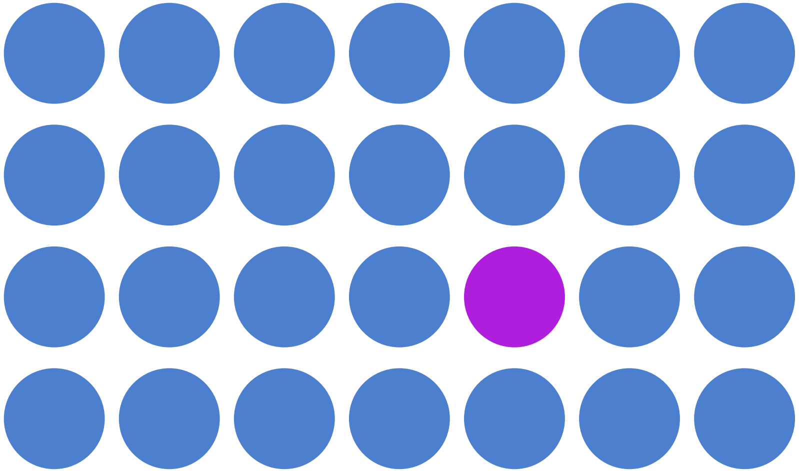
- You have a Where's Waldo book.
- You found Waldo and want to prove it to me.
- But you don't want to show me where Waldo is.

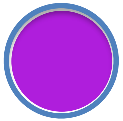
Where's Waldo?

The Classic Example

The Zero-Knowledge-Solution:

1. Take a large piece of cardboard
2. Cut a small hole in it
3. Place cardboard over the page
4. Position the hole exactly over Waldo
5. Show me Waldo through the hole





Why This Works?

Analyzing the Waldo Proof

Key Insights:

- **Verification:** I can see Waldo through the hole
- **No Information Leak:** The cardboard hides all context
- **Repeatability:** We can do this for any Waldo puzzle

But this isn’t a formal ZKP:

- **No Interaction:** Verifier can’t issue a challenge
- **Not mathematically verifiable:** Requires physical presence
- **Not simulatable:** can't create fake transcripts

Understanding Proofs

The Problem Domain

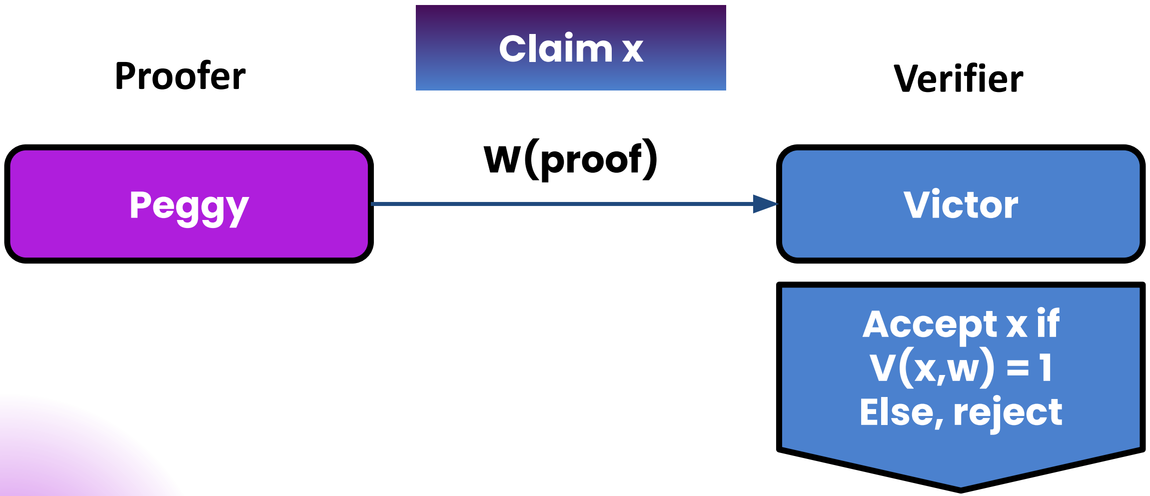
Decision Problems:

- Decision problems have binary answers: Yes, No
- “Is it raining outside of the venue?”
- “Is X a prime number?”
- “Given x and y, is x a multiple of y?”

Defines a whole class of Problems we can solve using computers.

Understanding Proofs

A classical proof



Understanding Proofs

Computational Complexity

NP: Nondeterministic Polynomial Time

- Can be hard to solve.
- Easy to verify
- Like a Sudoku
- Or factorization of primes

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

Defines a whole class of Problems we can solve using computers.

Understanding Proofs

A classical decision proof

Proofer

Claim:
N is product of 2 primes

Verifier

$\text{Peggy} \xrightarrow{W=\{p,q\}} \text{Victor}$

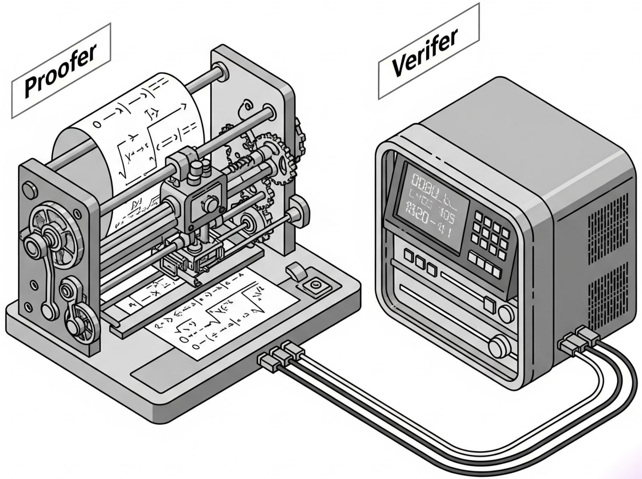
V(x,w): (pq)/N
Is 1 if pq=N
Accept if 1, else reject

Understanding Proofs

Summary

For a decision problem Q(x) in NP:

- **Complete:** If Q(x)=YES, than there is a proof w such that V(x,w) = 1
- **Sound:** If Q(x) = No, than for all proofs w, V(x,w) = 0



Interactive Proofs
Interaction and Randomness

Scenario

- You can see colors.
- Everybody else can't.
- How can you proof that you do see colors?

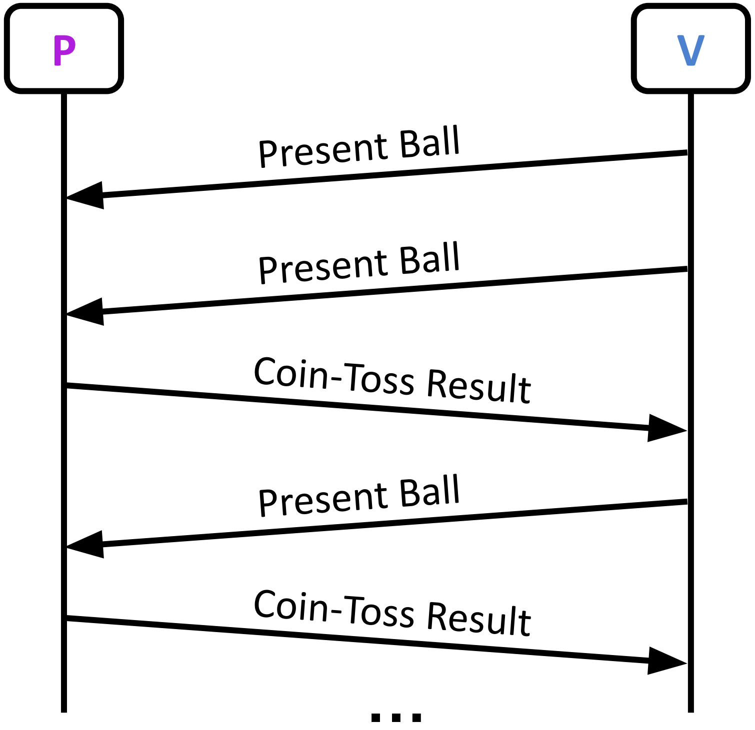
Setup

- Claim is: "I can see colors" or "I can see more than you"
- You are the proofer
- Some representative of the others is the verifier
- You have two balls of different color

Interactive Proofs
Interaction and Randomness

Proof System (Protocol)

1. V takes the balls and reveals one to you
2. V tosses a coin in private and shows you a ball based on the result:
   a. Heads: show other ball
   b. Tails: show same ball
3. P tells V the result of the coin toss.
4. Repeat



Interactive Proofs
Summary

- Instead of proving a statement directly we proved that we know something the other party don’t
- We don’t proof a 100% but with a certain confidence
- For this we introduced repeated interaction and randomness

- **Complete:** If Q(x)=YES, than there is a proof w such that V(x,w) = 1
- **Sound:** If Q(x) = No, than for all proofs w, V(x,w) = 0 except with negl. Probability
- Negl < 1/polynomial(|x|)
- Example: (½)^k where k is no rounds

Zero -Knowledge-Proofs

Let's put it all together

- **Complete:** If Q(x)=YES, than there is a proof w such that V(x,w) = 1

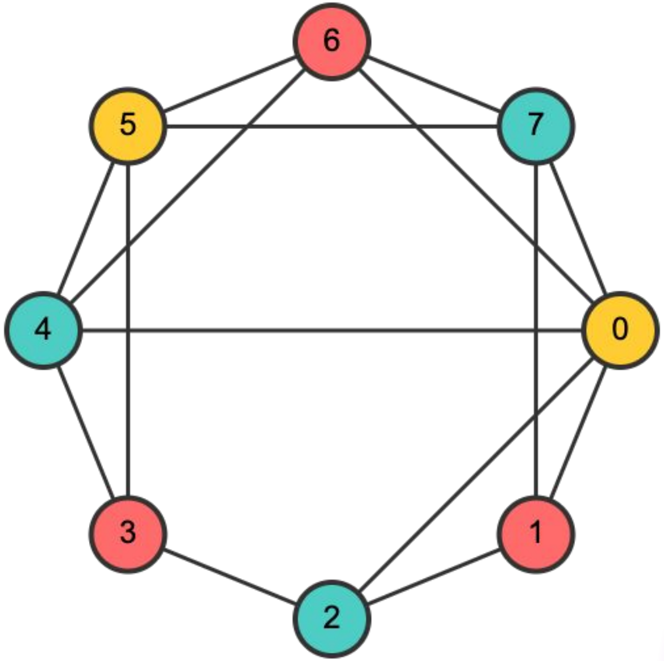
- **Sound:** If Q(x) = No, than for all proofs w, V(x,w) = 0 except with negligible Probability

- **Zero-Knowledge:** If Q(x) = YES, than V does not learn anything other than the fact that the statement is true

Zero-Knowledge-Proof
For the 3-Coloring Problem

Scenario

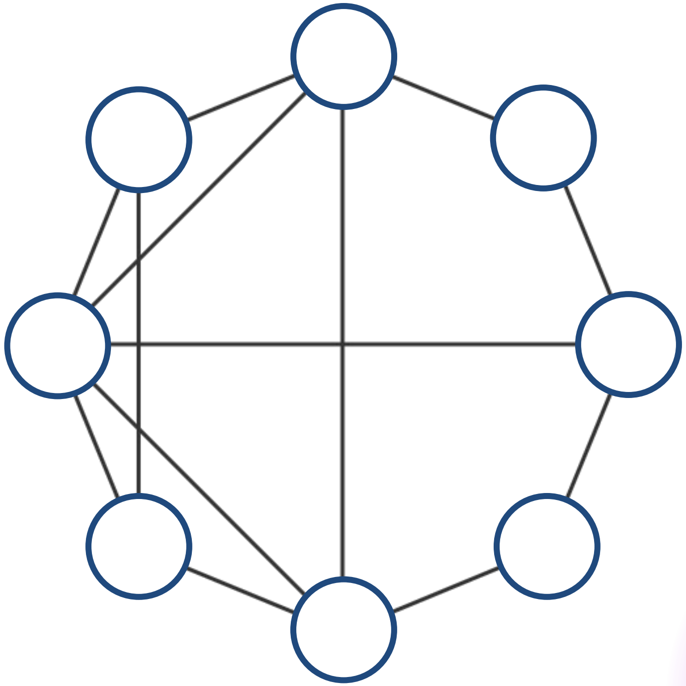
- You have a graph
- You have 3 colors
- The nodes of the graph shall be colored
- Can the whole graph be colored such, that no two adjacent nodes have the same color?
- Claim is: “Graph is 3-colorable”



Zero-Knowledge-Proof

For the 3-Coloring Problem

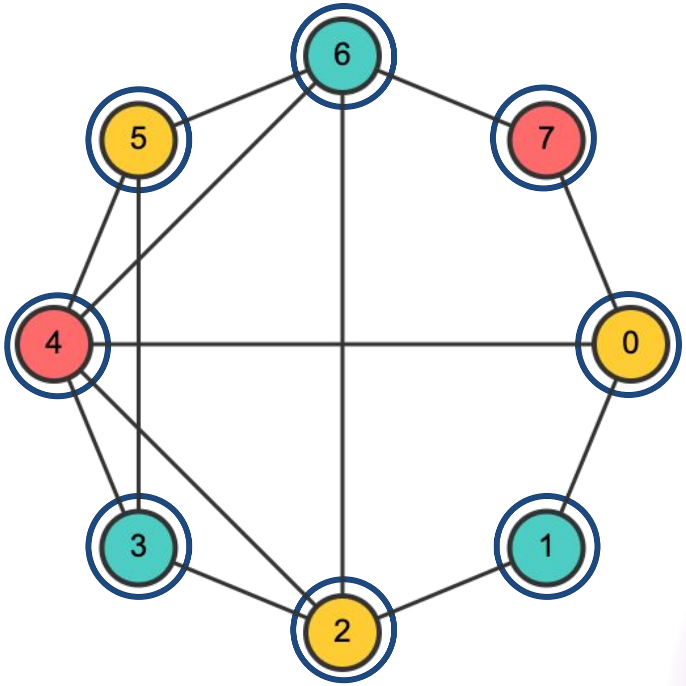
Protocol v1



Zero-Knowledge-Proof
For the 3-Coloring Problem

Protocol v1

1. Prover: Send Verifier our solution as proof

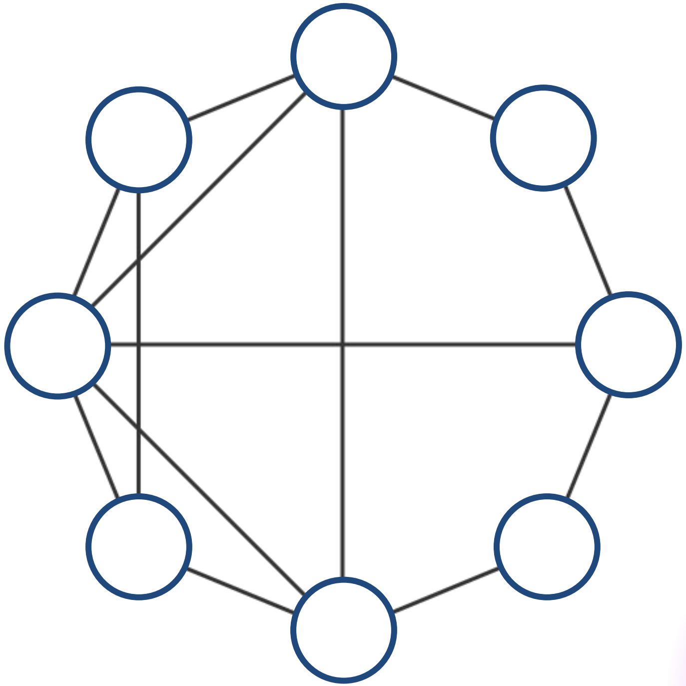


Zero-Knowledge-Proof

For the 3-Coloring Problem

Protocol v2

1. Verifier: Roll a n-sided dice and ask to reveal the corresponding edge as a sample
2. Prover: Send tuple

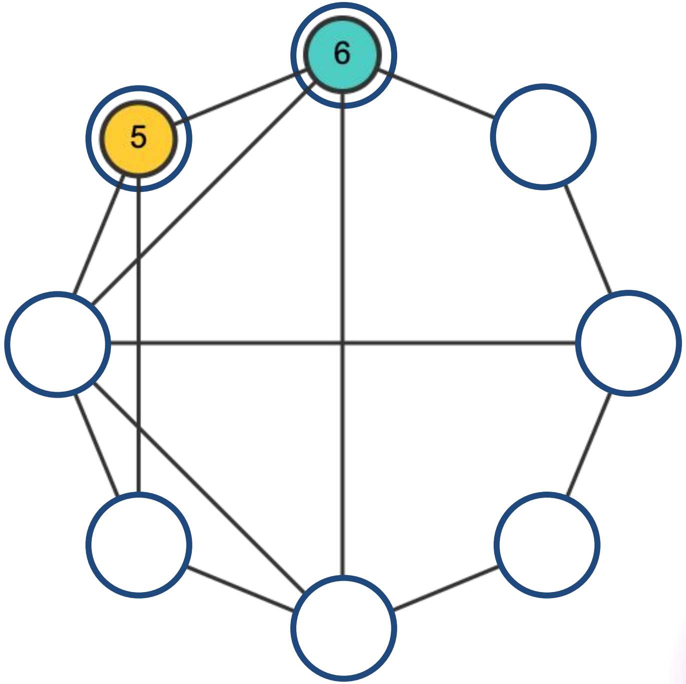


Zero-Knowledge-Proof

For the 3-Coloring Problem

Protocol v2

1. Verifier: Roll a n-sided dice and ask to reveal the corresponding edge as a sample
2. Prover: Send tuple
3. Verifier: Check that c_n1 != c_n2
4. Repeat



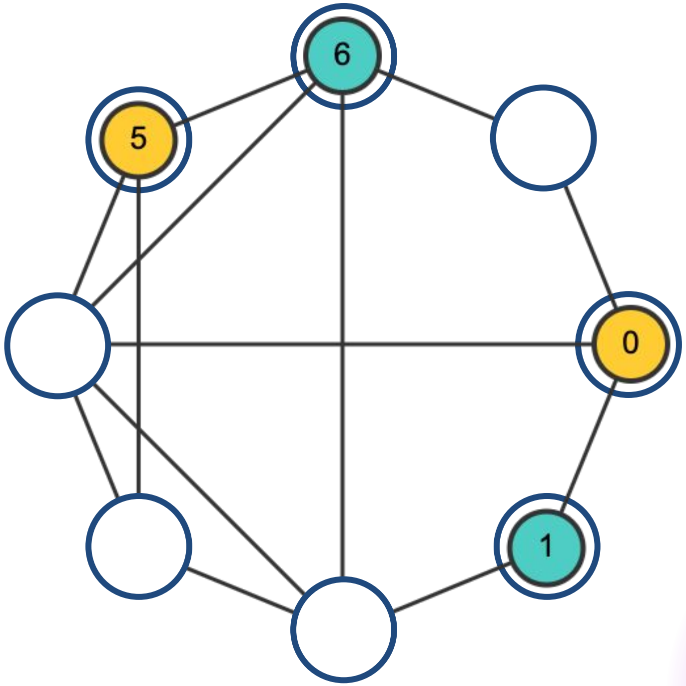
Soundness error is (1-1/|E|)^k, E is edges, k is rounds

(1-1/|E|)^(n|E|) approx e^(-n) is negligible

Zero-Knowledge-Proof
For the 3-Coloring Problem

Protocol v2

1. Verifier: Roll a n-sided dice and ask to reveal the corresponding edge as a sample
2. Prover: Send tuple
3. Verifier: Check that c_n1 != c_n2
4. Repeat

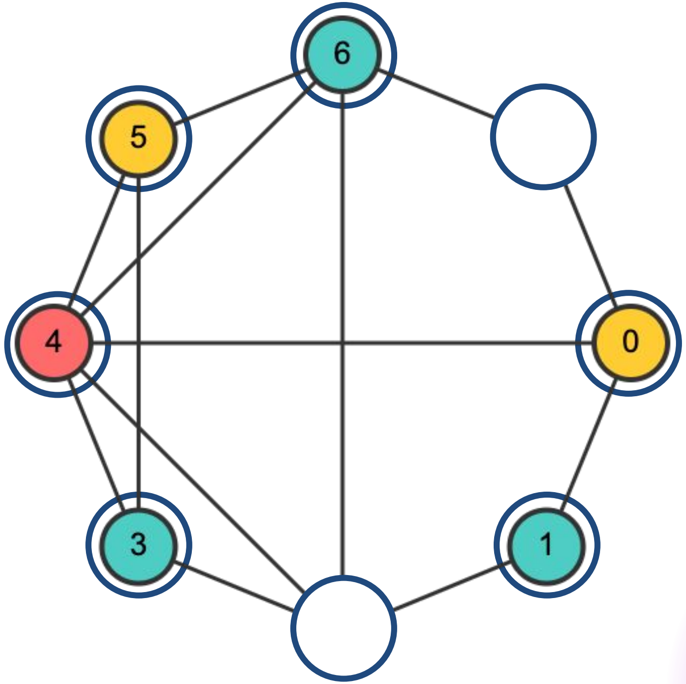


Zero-Knowledge-Proof

For the 3-Coloring Problem

Protocol v2

1. Verifier: Roll a n-sided dice and ask to reveal the corresponding edge as a sample
2. Prover: Send tuple
3. Verifier: Check that c_n1 != c_n2
4. Repeat

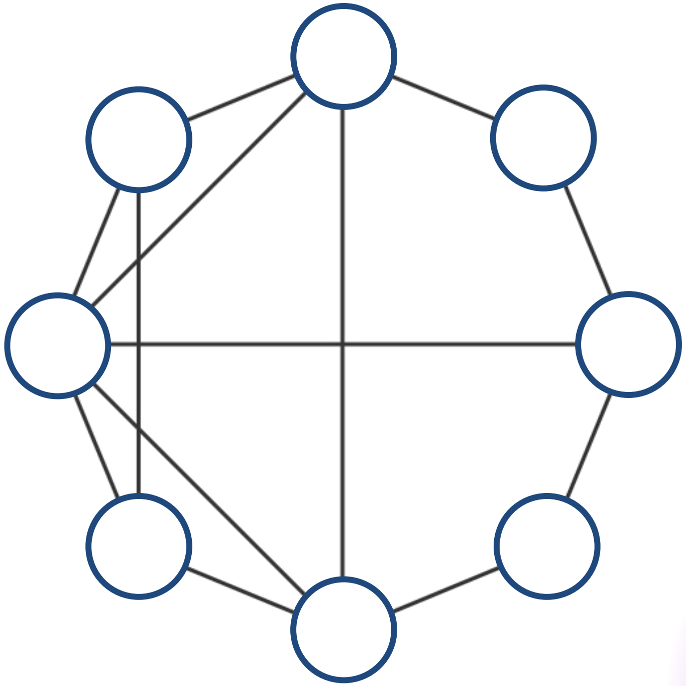


Zero-Knowledge-Proof

For the 3-Coloring Problem

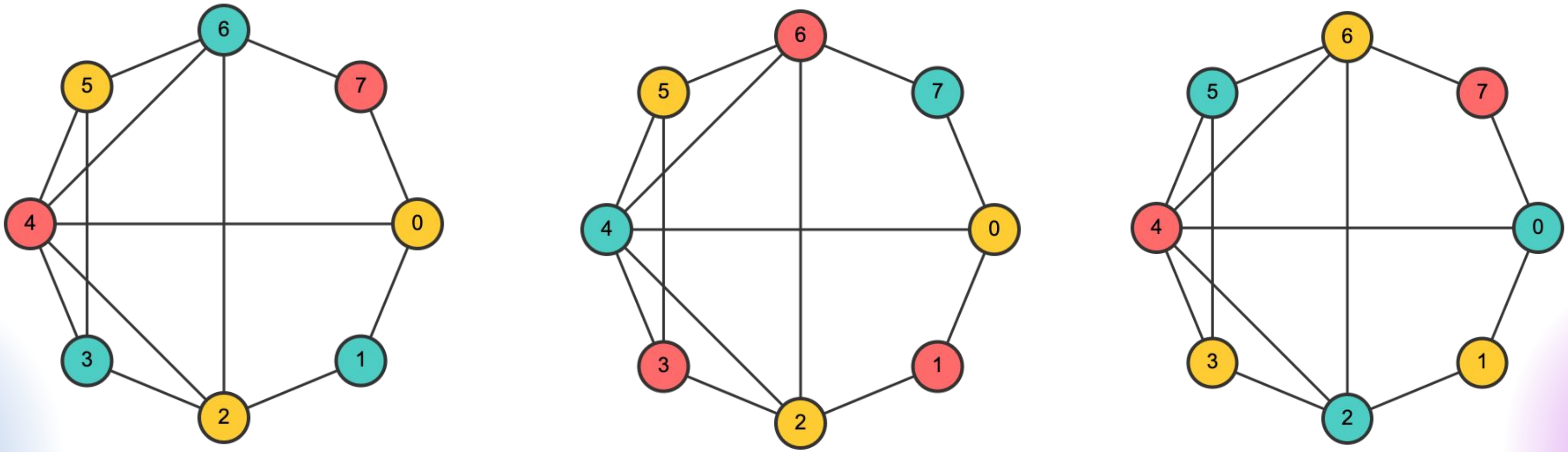
Protocol v3

1. Verifier: Roll a n-sided dice and ask to reveal the corresponding edge as a sample
2. Prover: Send a permutation of the tuple
3. Verifier: Check that c_n1 != c_n2
4. Repeat



Zero-Knowledge-Proof

For the 3-Coloring Problem



Zero-Knowledge-Proof

For the 3-Coloring Problem

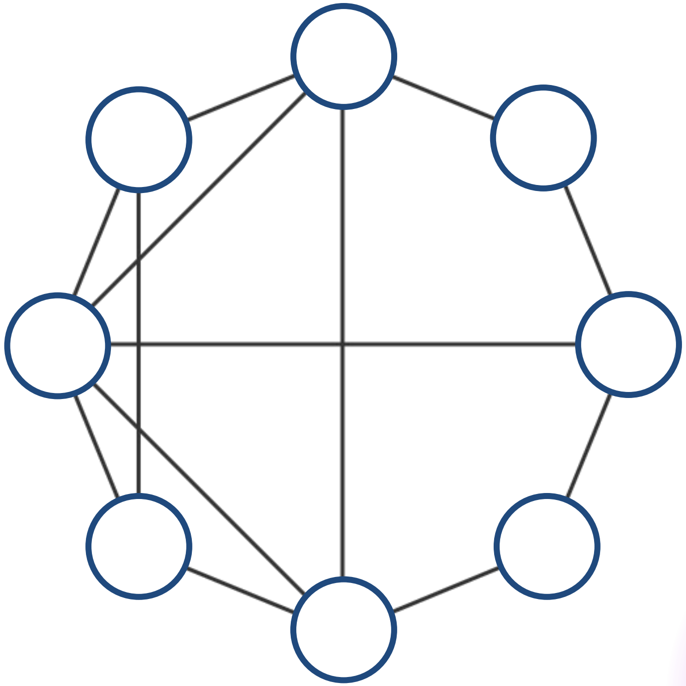
https://nepet.github.io/btcpp-3-color-zkp/

Zero-Knowledge-Proof

For the 3-Coloring Problem

Protocol 3

1. Verifier: Roll a n-sided dice and ask to reveal the corresponding edge as a sample
2. Prover: Send a permutation of the tuple
3. Verifier: Check that c_n1 != c_n2
4. Repeat

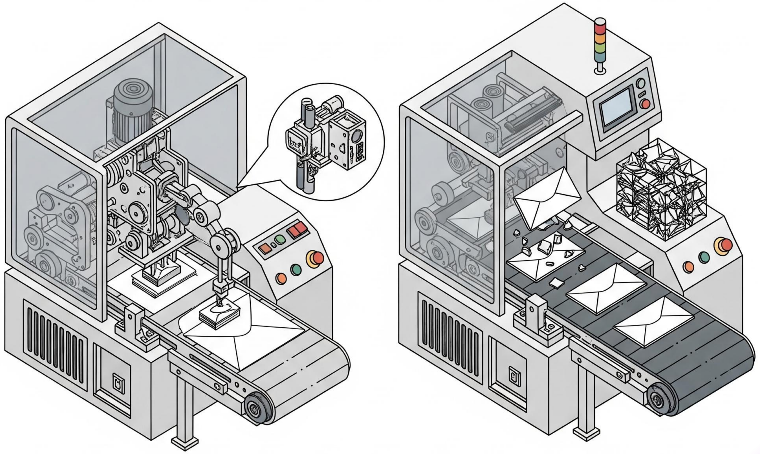


Zero-Knowledge-Proof

For the 3-Coloring Problem

Commitment

1. Prover: Randomly permute the 3-colors
2. Prover: Calculate for every node
   COM(id, c, r) = SHA256(id || c || r)
3. Prover: Send commitments
4. Verifier: When checking c_n1 != c_n2, also
   verify commitments for n1 and n2

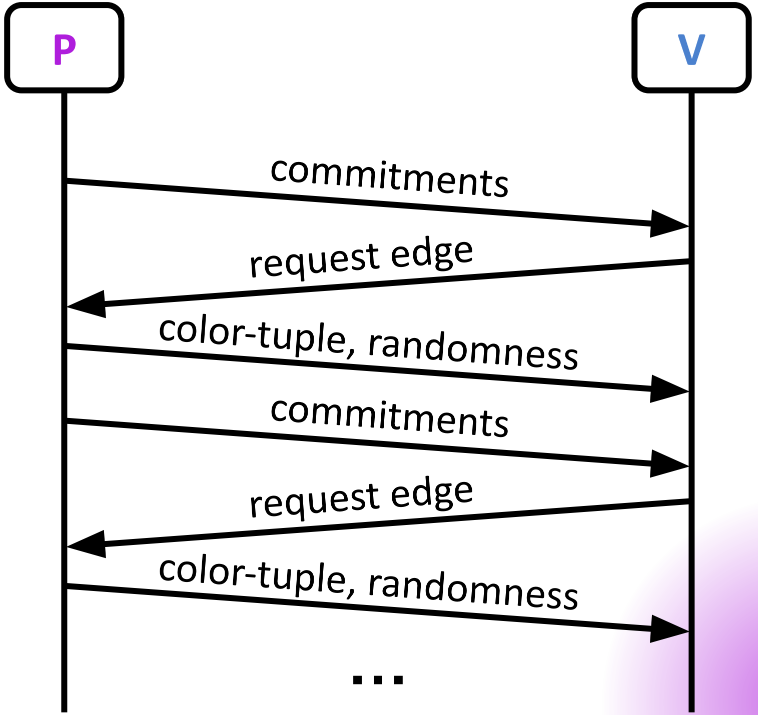


Zero-Knowledge-Proof

For the 3-Coloring Problem

Protocol 4

1. Prover: Permute solution
2. Prover: Send commitment
3. Verifier: Roll a n-sided dice and ask to reveal the corresponding edge as a sample
4. Prover: tuple and randomness
5. Verifier: Check that c_n1 != c_n2 + coms n1 and n2
6. Repeat



Zero-Knowledge-Proof

For the 3-Coloring Problem

https://nepet.github.io/btcpp-3-color-zkp/

Zero-Knowledge-Proof

For Every Problem in NP

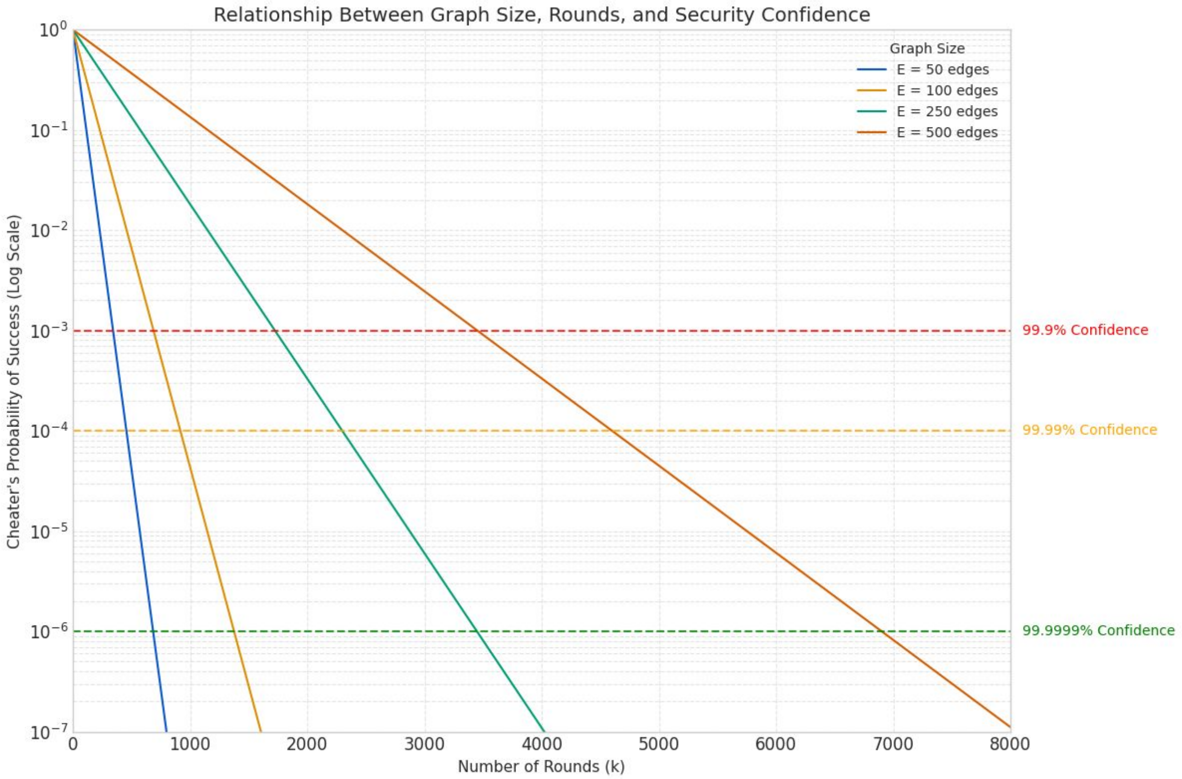
- We constructed a perfect ZKP
- No information is leaked whatsoever
- But it is impractical for a lot of Applications
- Huge amount of round-trips, Huge amount of data
- But Proof by Oded Goldreich, Silvio Micali et.al in 1986 laid the foundation
- Core of the proof is 3-Coloring

bitcoin++
Istanbul

twitter: @nepetonium
github.com/nepet

Thank You
Teşekkürler

https://github.com/nepet/btcpp-3-color-zkp


Relationship Between Graph Size, Rounds, and Security Confidence
Graph Size
E = 50 edges
E = 100 edges
E = 250 edges
E = 500 edges
99.9% Confidence
99.99% Confidence
99.9999% Confidence
Cheater's Probability of Success (Log Scale)
Number of Rounds (k)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
1000
2000
3000
4000
5000
6000
7000
8000