Lecture 27
Non-Interactive Zero Knowledge Proof

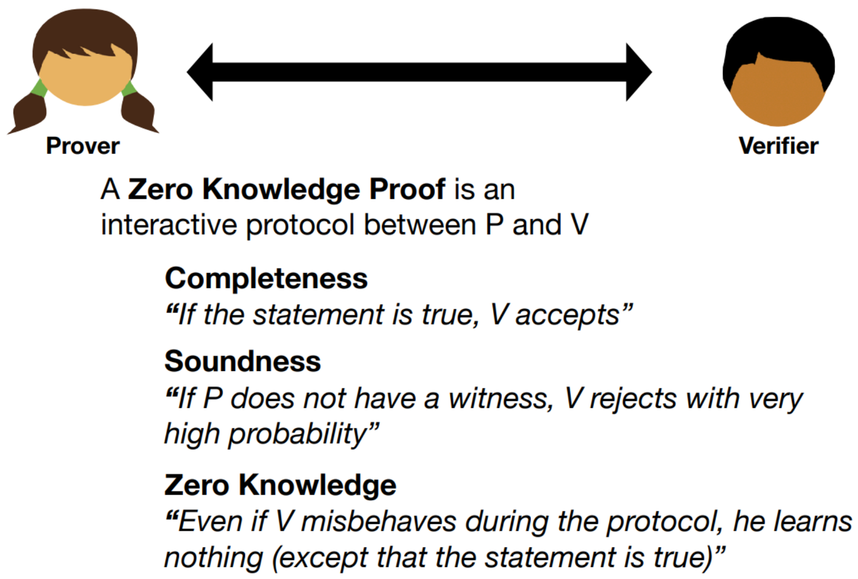
Lecturer: Professor David Heath
Scribe: Aditya Perswal

December 7, 2023

Today's Objectives

- Examine the round complexity of ZK proofs
- Construct a non-interactive ZK-proof system
- Construct post-quantum signatures
- Show NIZK does not exist in the standard model

1. Introduction to Zero-Knowledge Proofs


Prover
Verifier

A **Zero Knowledge Proof** is an interactive protocol between P and V

Completeness
"*If the statement is true, V accepts"*

Soundness
"*If P does not have a witness, V rejects with very high probability"*

Zero Knowledge
"*Even if V misbehaves during the protocol, he learns nothing (except that the statement is true)"*

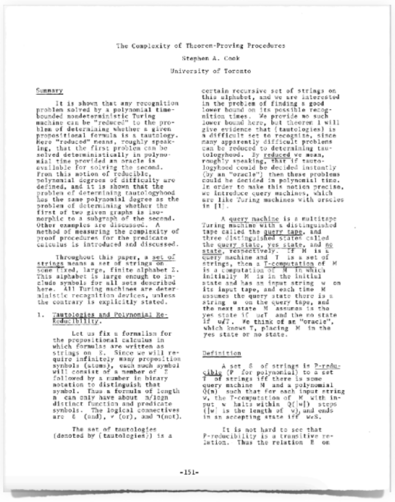
Zero-knowledge proofs (ZKPs) are a revolutionary cryptographic method, ensuring that one party (the prover) can prove to another party (the verifier) that they know some information (i.e value x), without conveying any piece of that information apart from the fact that they know the information (i.e value x).

Basic Concepts

- **Prover (P)**: The person/party claiming to have knowledge or information, with the aim to convince the verifier of this possession of information without revealing the information itself.
- **Verifier (V)**: The person/party that sends challenges to the prover's claim, with the goal to validate the authenticity of the prover's statement.
- **Agreed Statement**: Before the proof begins, both parties agree on a certain statement. In the context of graph theory, this is typically a graph without any assigned colors. The prover then tries to convince the verifier that they know a valid 3-coloring of the graph without revealing the coloring itself.

2. Graph 3-Coloring Protocol

Graph coloring has been a problem studied for over a hundred years. Zero Knowledge Proofs are able to take advantage of this graph theory to establish a protocol in which, a prover attempts to convince a verifier that a given graph can be colored using just three colors, such that no two adjacent nodes share the same color, without revealing the actual coloring. The ability of this being even possible was due to a research paper released by the University of Toronto.



The Complexity of Theorem-Proving Procedures

Stephen A. Cook

University of Toronto

Summary

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

Throughout this paper, a set of strings means a set of strings on some fixed, large, finite alphabet Σ. This alphabet is large enough to include symbols for all sets described here. All Turing machines are deterministic recognition devices, unless the contrary is explicitly stated.

1. Tautologies and Polynomial Re-Reducibility.

Let us fix a formalism for the propositional calculus in which formulas are written as strings on Σ. Since we will require infinitely many proposition symbols (atoms), each such symbol will consist of a member of Σ followed by a number in binary notation to distinguish that symbol. Thus a formula of length n can only have about $n/\log n$ distinct function and predicate symbols. The logical connectives are $\&$ (and), $\vee$ (or), and $\neg$(not).

The set of tautologies (denoted by {tautologies}) is a certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By reduced we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].

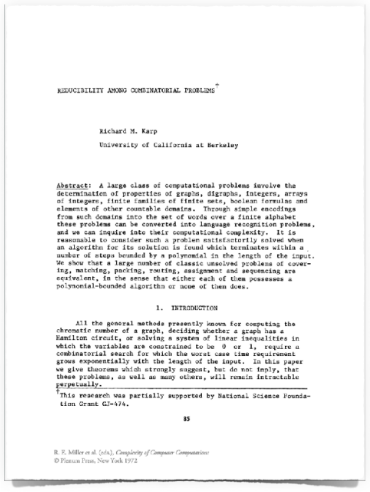
A query machine is a multitape Turing machine with a distinguished tape called the query tape, and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a set of strings, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and the next state M assumes is the yes state if $u \in T$ and the no state if $u \notin T$. We think of an "oracle", which knows T, placing M in the yes state or no state.

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps ($|w|$ is the length of w), and ends in an accepting state iff $w \in S$.

It is not hard to see that P-reducibility is a transitive relation. Thus the relation E on

-151-



REDUCIBILITY AMONG COMBINATORIAL PROBLEMS†

Richard M. Karp

University of California at Berkeley

Abstract: A large class of computational problems involve the determination of properties of graphs, digraphs, integers, arrays of integers, finite families of finite sets, boolean formulas and elements of other countable domains. Through simple encodings from such domains into the set of words over a finite alphabet these problems can be converted into language recognition problems, and we can inquire into their computational complexity. It is reasonable to consider such a problem satisfactorily solved when an algorithm for its solution is found which terminates within a number of steps bounded by a polynomial in the length of the input. We show that a large number of classic unsolved problems of covering, matching, packing, routing, assignment and sequencing are equivalent, in the sense that either each of them possesses a polynomial-bounded algorithm or none of them does.

1. INTRODUCTION

All the general methods presently known for computing the chromatic number of a graph, deciding whether a graph has a Hamilton circuit, or solving a system of linear inequalities in which the variables are constrained to be 0 or 1, require a combinatorial search for which the worst case time requirement grows exponentially with the length of the input. In this paper we give theorems which strongly suggest, but do not imply, that these problems, as well as many others, will remain intractable perpetually.

†This research was partially supported by National Science Foundation Grant GJ-474.

85

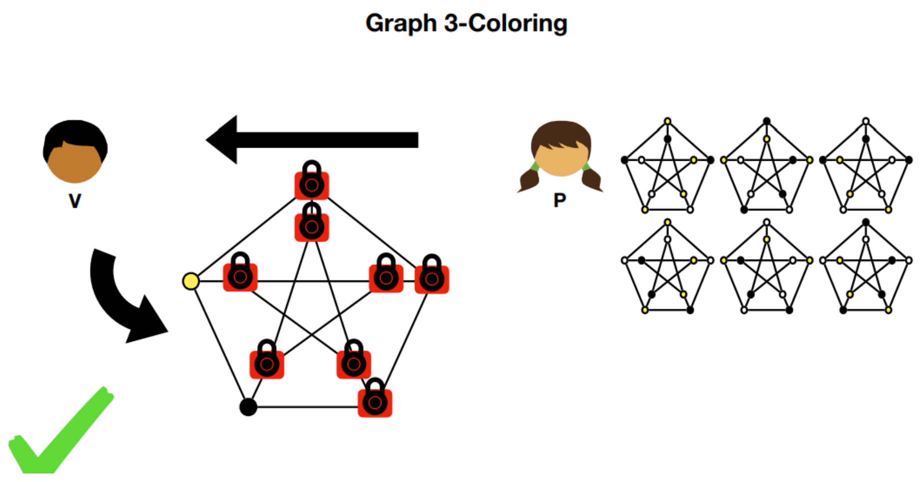
R. E. Miller et al. (eds.), *Complexity of Computer Computations*
© Plenum Press, New York 1972

Protocol Flow

- **Prover Commits to a Coloring**: Initially, the prover selects a valid 3-coloring of the graph and commits to it without revealing the actual colors to the verifier. This commitment is the same as sealing the coloring in an envelope; the prover knows what's inside, but the verifier doesn't.

- **Verifier Issues a Challenge**: After the commitment, the verifier, issues a challenge. This would be a request for the prover to unveil the color of a particular node or to verify the colors of two adjacent nodes.

- **Prover Reveals Part of the Commitment**: In response to the verifier's challenge, the prover unveils part of the committed coloring that was chosen, allowing the prover to not show the whole graph.



Outcome:

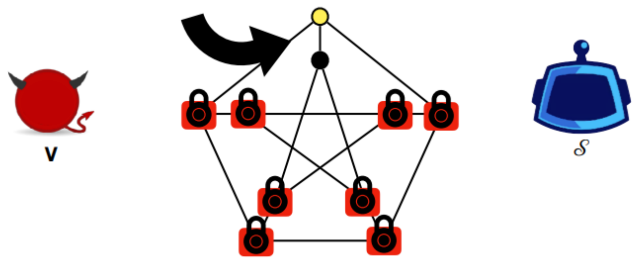
- **Completeness**: If the statement is true then the verifier accepts.
- **Soundness**: If the prover does not have a witness (correct coloring or information) then the verifier rejects with a very high probability. This is because the prover won't consistently respond correctly to the verifier's challenges, since the verifier issues multiple challenges.
- **Zero Knowledge**: If the verifier misbehaves during the protocol, they would learn nothing, except the statement being true. They learn nothing beyond the validity of the statement, ensuring the prover's secret remains safe.

With the foundation set, it's time to delve into the role of the simulator in zero-knowledge proofs. The simulator plays a pivotal role, ensuring the zero-knowledge property remains intact even in the face of adversarial verifiers.

3. Role of the Simulator in Zero-Knowledge Proofs

A critical aspect of Zero-Knowledge Proofs is the use of a simulator. The simulator is used to show that any information the verifier might get from challenging the prover could have been produced without the prover's secret knowledge.

Graph 3-Coloring
Zero Knowledge



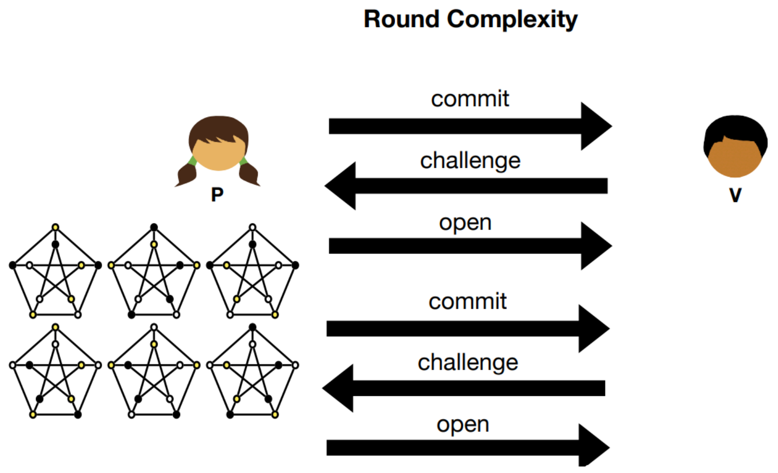
Adversarial V will soon (in polynomial tries) choose the chosen edge

After polynomial iterations, the proof interaction will terminate

- **Acts like the Prover**: The simulator acts like the prover, mimics the actions of the prover in terms of generating commits and responses to the challenges sent by the verifier. However, the simulator has no clue what the information is and as such is making random variations of the 3 color graph.
- **Interacts with Possibly Adversarial Verifier**: The simulator is trying to show that even if the verifier was malicious and tried to deviate from the protocol to extract information, they wouldn't gain an advantage. The simulator is showing that the output of interactions between a simulator and a prover is indistinguishable.
- **Probability: 1/e (where e is Euler's number)**: For instance, in some ZKP protocols, the simulator might have a 1/e chance in guessing the verifier's challenge correctly.

4. Enhancing Soundness in Zero-Knowledge Proofs

In Zero-Knowledge Proofs (ZKPs), soundness is a crucial property. It guarantees that a dishonest prover, cannot convince the verifier of the truth of their statement with any significant probability.



- **Sequentially Repeat Sigma Protocols**: One approach to enhance soundness is to simply repeat the Sigma Protocol (Commit - Challenge - Open) multiple times in a sequential manner. Each repetition starts from the beginning, and if the prover is dishonest, they're likely to be caught out in one of these repetitions.

However, this can take a large amount of time due to the number of times a verifier will challenge the prover. Hence a better method has come to place.

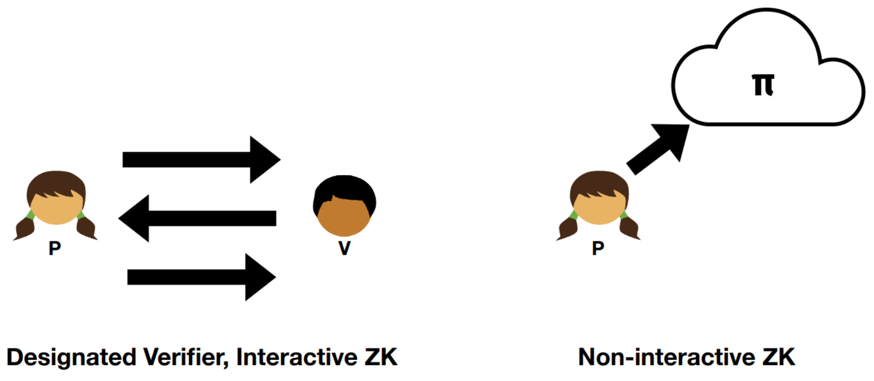
Correct Methodology to Enhance Soundness:

- **Commit to Challenges**: Before the protocol begins, the verifier commits the issues they will challenge making it so they can't change the challenges based on the prover's commits.
- **Commit to All Iterations**: The prover commits to all iterations of challenges committed.
- **Open Challenged Iterations**: The prover then opens all iterations of challenges committed only showing the information needed for verification.

As you may be able to tell, this can become problematic for a simulator as it would require the simulator to get the right graph coloring for every committed challenge rather than just 1. This would mean even if the simulator got it correct 99 times out of 100, it would not be a success because it needs a 100/100 perfect coloring occurrence, which is an extremely low probability. This is where Non-Interactive Zero-Knowledge Proofs come to save the day.

5. Non-Interactive Zero-Knowledge (NIZK) Proofs

Non-Interactive Zero-Knowledge Proofs (NIZKs) provide a way to prove knowledge of a statement without the need for interaction between the prover and verifier. Unlike traditional ZKPs, where multiple rounds of interaction occur, NIZKs prove in one go.



Basics of NIZK:

- **Challenges are Generated Through a Random Oracle**: NIZKs use a random oracle, typically modeled by a hash function, to generate challenges. This allows for no interaction to occur as the challenges are deterministically generated from the commitments.
- **Construction**:
  - **Hash(commit) = challenges**: The prover hashes their commitment to produce the challenges.
  - **Prove by Showing Commit and Openings**: The prover then shows both the initial commitment and the responses to the challenges. The verifier, knowing the hash function, can independently verify the proof by hashing the commitment and checking the responses.
- **Role of Random Oracles**:
  - **Essential for Constructing NIZK**: The use of random oracles is necessary for challenge generation and to allow the proof construction to be non-interactive.
  - **Impossibility in the Standard Model**: NIZKs can't exist in the standard model of cryptography, since they need random oracles which can't exist in the standard model.

6. Questions

1. **Are there restrictions on the simulator in zero-knowledge proof, such as Polynomial Time?**

   - In zero-knowledge proofs, the simulator is an essential construct. Its role is to generate transcripts of interactions that look indistinguishable from real interactions, but without having the secret (witness). One key restriction is that the simulator should operate in polynomial time to ensure the feasibility and practicality of the system.
   - **Polynomial-time Restriction**: In computational theory, an algorithm is said to run in polynomial time if its runtime grows at most as some polynomial in the size of the input.

2. **Can you make a zero-knowledge system without soundness or completeness?**

   - Without soundness or completeness, the system would either accept false proofs or reject true proofs, respectively.
   - **Completeness**: This ensures that an honest prover can convince an honest verifier of the truth of a statement.
   - **Soundness**: This guarantees that a dishonest prover cannot convince an honest verifier of a false statement.
   - **Zero-knowledge**: This ensures the prover doesn't reveal any additional information beyond the validity of the statement.

3. **Is the verifier deterministic?**

   - Verifiers in zero-knowledge proofs can be either deterministic or randomized. However, when discussing simulators and rewind techniques, the verifier's randomness can be thought of as coming from a preset tape of random bits.
   - **Randomized Turing Machine**: A machine that can make decisions based on random bits, foundation of randomized algorithms.

4. **Is the simulator extremely lucky?**

   - The simulator's design is such that it can produce valid-looking interactions without knowing the secret. This might seem like "luck," but it's the outcome of careful construction.

5. **So every time there is a rewind, the verifier commits the same challenges?**

   - Yes, if the simulator controls the verifier's randomness (e.g., by controlling a random tape), rewinding leads to the verifier producing the same challenges.

6. **Why can't the prover cheat in non-interactive zero-knowledge?**

    - In non-interactive zero-knowledge proofs, the challenge typically comes from a hash function acting as a random oracle. Cheating would require predicting the output, which is computationally hard.
    - **Random Oracle Model**: A theoretical model where hash functions behave as truly random functions.

7. **In the non-interactive zero-knowledge, is it zero-knowledge only because of the random oracle?**

    - Many non-interactive zero-knowledge proofs rely on the random oracle model for their zero-knowledge property.
    - **Random Oracle Assumption**: An assumption that a hash function behaves unpredictably and uniformly.

Resources Used:

- David Heath. CS 407 Cryptography Lectures. 2023.
- Jonathan Katz and Yehuda Lindell. Introduction to Modern Cryptography. 2007.