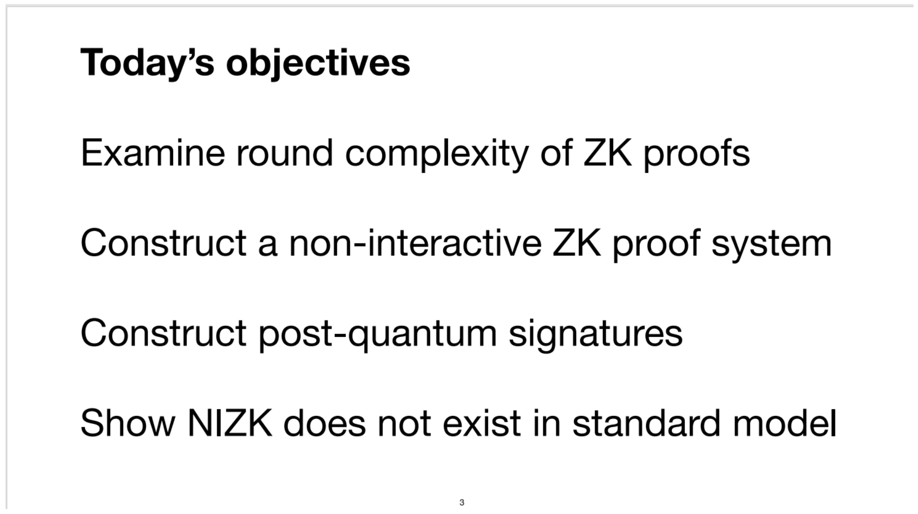
Lecture 24 Non-Interactive Zero Knowledge Proofs

Lecturer: Professor David Heath
Scribe: Samvit Dedhia

November 7, 2023

This lecture focuses on a problem with complexity of the Zero Knowledge Proof protocol and introduces a potential solution by using Non-Interactive ZK proofs.



Today's objectives

Examine round complexity of ZK proofs

Construct a non-interactive ZK proof system

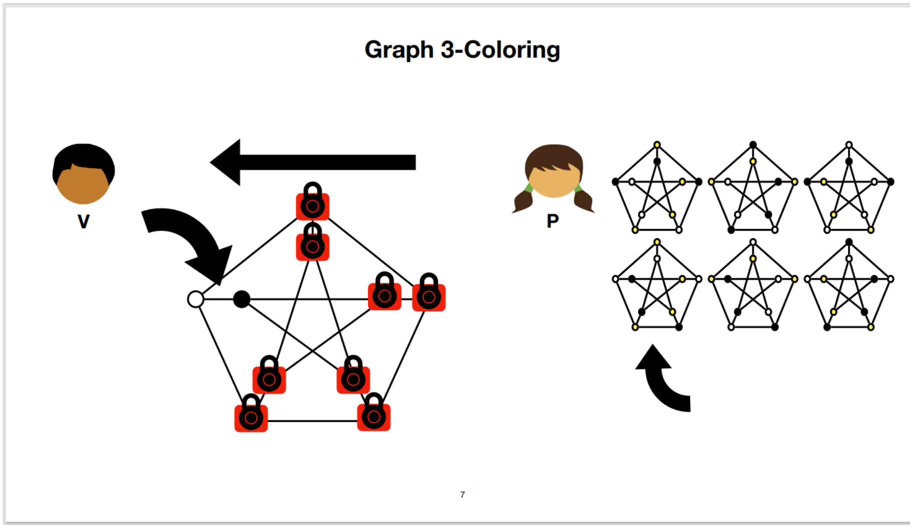
Construct post-quantum signatures

Show NIZK does not exist in standard model

3

Last lecture, we learned about the details of Zero Knowledge proofs, including the protocol used between the prover and the verifier. Specifically, we learned about the 3 crucial properties of Zero Knowledge proofs, and how to prove that a it indeed is a zero knowledge proof by using indistinguishability of the prover and the simulator. In this lecture, we review the properties of Zero Knowledge Proofs, examine their round complexity, including issues with the protocol when used at scale. The lecture introduces the concept of round complexity, elucidating the efficiency and interactions between the prover and verifier in a protocol. Moreover, we transition into an advanced segment on non-interactive zero-knowledge proofs, where traditional back-and-forth communication is eschewed for a more streamlined approach. A potential solution to the problem of complexity by using a non-interactive protocol system for the ZK proof system. The lecture also touches upon their real-world applications, notably in quantum signature schemes, showcasing their significance in modern cryptographic applications.

PROPERTIES OF ZERO KNOWLEDGE PROOFS:



In-order for a proof to be considered Zero Knowledge, it must follow the three basic rules: Completeness, Soundness, and most importantly, Zero Knowledge.

Completeness:

Definition: If the statement is true and both the prover and the verifier follow the protocol correctly, the verifier will be convinced of the statement's truth.

Graph 3-Coloring Example: Imagine a graph that can be properly 3-colored. If the prover knows a valid 3-coloring of this graph and wants to convince the verifier of this without revealing the actual coloring, the verifier will be convinced after the protocol's completion, assuming both parties follow it correctly.

Soundness:

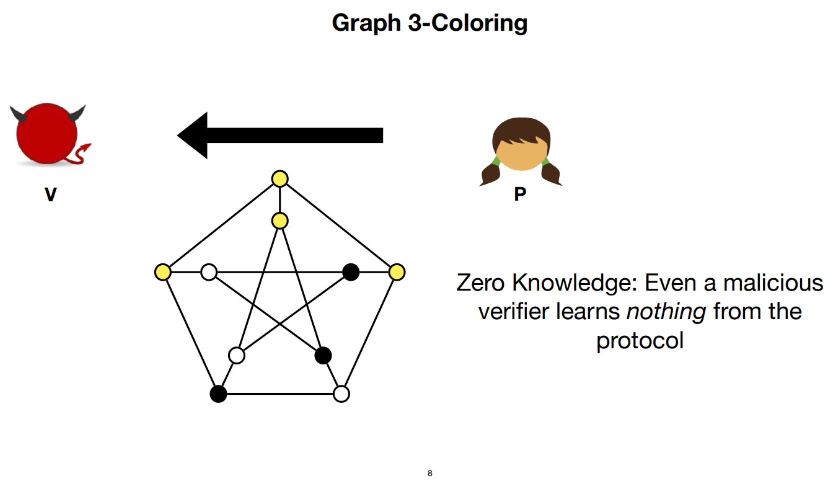
Definition: If the statement is false, a cheating prover cannot convince the verifier that the statement is true, except with negligible probability.

Graph 3-Coloring Example: If a graph is not 3-colorable, a dishonest prover will have an almost impossible task of trying to convince the verifier that it is.

Zero Knowledge:

Definition: If the statement is true, the verifier learns nothing beyond the validity of the statement. The concept of a ”simulator” is crucial here. For the proof to be zero-knowledge, there must exist a simulator that can produce a conversation transcript that looks just like a real interaction between the prover and the verifier, without knowing the prover’s secret. Crucially, this simulator has the power to ”reset” the verifier to its initial state and run the protocol multiple times if needed.

Graph 3-Coloring Example: Assume the prover knows a valid 3-coloring of a graph. During the ZKP interaction for 3-coloring, the verifier becomes convinced but learns nothing about the specific coloring.


Graph 3-Coloring
V
P
Zero Knowledge: Even a malicious verifier learns *nothing* from the protocol
8

Simulator

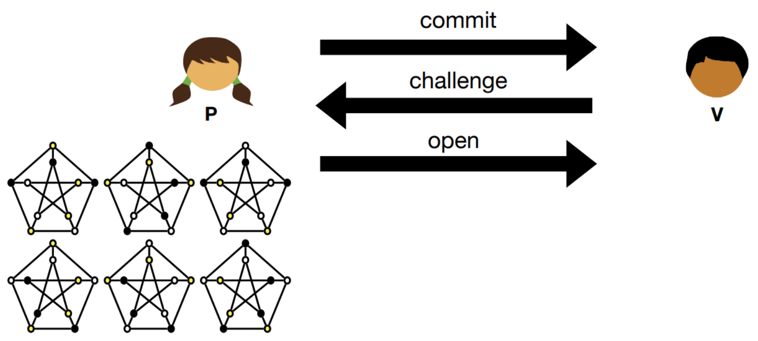
The simulator doesn’t know the actual 3-coloring but wants to produce a transcript that looks like a real interaction. The power to reset the verifier is crucial here. If, during the simulated interaction, the verifier asks a question that the simulator cannot answer without knowing the secret, the simulator can ”reset” the verifier and start the protocol over, as if the question was never asked. This way, the simulator can keep resetting and rerunning the protocol until it gets a sequence of questions from the verifier that it can answer, producing a transcript that looks like a real interaction.

To an external observer, this simulated transcript is indistinguishable from a genuine interaction between the prover and the verifier. This ensures that the verifier learns nothing more than the fact that the graph is 3-colorable, which is the essence of zero-knowledge.

In essence, the resettable simulator allows us to strengthen the zero-knowledge property by showing that even if the verifier can be reset and the protocol rerun many times, it still doesn't gain any additional information about the prover's secret.

Sigma Protocol:

The sigma protocol has 3 stages, Commit, Challenge, and Open. In the commit stage, the prover sends a solution to the verifier. In the challenge stage, the verifier asks the prover to reveal one part of the proof. And in the open stage, the prover sends that information to the verifier.



Commit Stage: The prover wants to convince the verifier that they know a valid 3-coloring of the graph without revealing the actual coloring. The prover randomly permutes (shuffles) the colors and commits to this permuted coloring by sending commitments for each vertex to the verifier. These commitments "lock" the color of each vertex without revealing it. The commitments can be generated using cryptographic techniques to ensure they can't be changed later without detection.

Challenge Stage: The verifier challenges the prover by randomly selecting an edge from the graph. The verifier's goal is to check if the two vertices connected by this edge have different colors.

Open Stage: The prover responds by opening the commitments for the two vertices of the selected edge, revealing the colors for those two vertices but not for any others. The verifier checks if the two revealed colors are different (ensuring the coloring rule is followed for that edge) and if they match the original commitments.

Round Complexity:

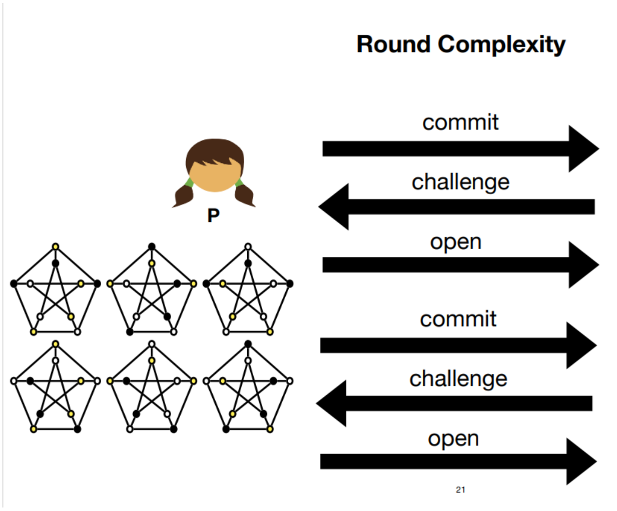
Roud complexity is the amount to interactions required between the prover and the verifier. The fewer the rounds, the more efficient the protocol is perceived to be.

Problems of the Standard ZK proof Protocol:

Since the simulator only send back one part of the solution and essentially hopes that the verifier chooses it, and if it doesn't, it reset's the tape of the verifier, the probability of the simulator getting that edge accurately is 1/—E—, therefore it would take the simulator an exponential number of tries to be able to "guess" the correct edge, and this makes the protocol of exponential round complexity.

Graph 3 Coloring Example of the Protocol:

Let's take the example of the graph 3 coloring problem to see why this protocol is of exponential complexity, with the example of the simulator.



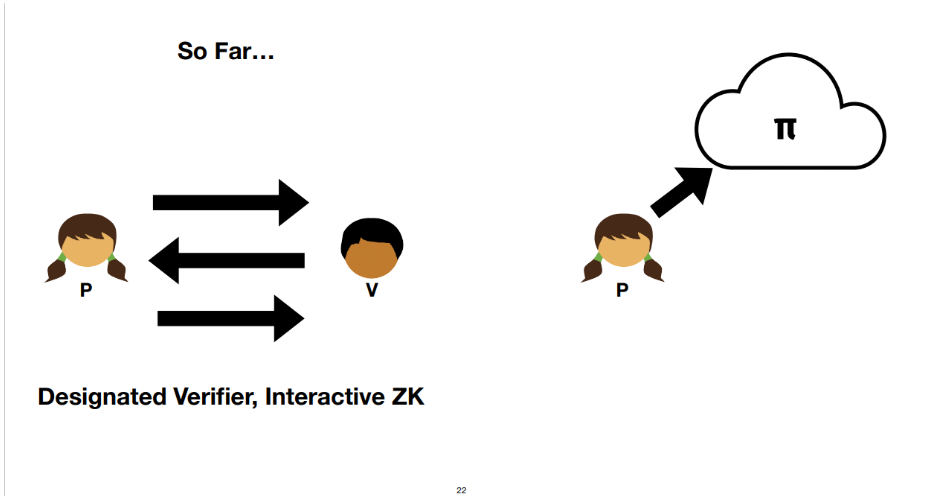
Since the simulator does not know the actual solution, and is only trying the "trick" the verifier into believing that it knows the solution, it picks a random edge, and colors it with 2 different colors and leaves all the other edges blank, and commit's the solution. When the verifier sends the challenge, if the simulator has sent the solution for that particular edge, it open's the solution, and the verifier believes the provided solution; however, if the simulator left that edge blank, it simply hits the reset button on verifier and receives a new challenge. It repeats this process until it receives a challenge for which it had committed a correct solution.

Since it has a 1/—E— probability of getting the correct solution for each round of verification, its round complexity is exponential in the number of verifications needed, or in other words, the number of times the verifier decides to use the sigma protocol.

Potential Solution to reduce the problem to an polynomial complexity

A simple fix to the protocol was discussed in class that seemed to have resolved the problem with exponential round complexity. Instead of the prover committing the solution first, and then the verifier challenging the prover, if the protocol was modified so that the verifier would commit to all the challenges first and then the prover would open the solution. And since the simulator has the ability to "rewind" the tape, to the last correct guess, and continue from there. Unlike the previous solution, where the simulator needs to make continuous correct guesses, if the challenges are committed first, each time a guess is incorrect, the simulator can rewind the tape.

We can Do better! Non-Interactive ZK Proofs:



In interactive/traditional zero-knowledge proofs, a prover and a verifier engage in a series of exchanges to convince the verifier of the truth of a statement without revealing any specific information about it. As we saw, they can have really large round complexities, and are unfavourable for most real-life applications. Enter Non-Interactive Zero-Knowledge (NIZK) proofs: these are protocols where the prover can generate a single proof, without any interaction, that convinces any verifier of the truth of the statement.

We can use our old friend, a random oracle to be able to create a NIZK protocol. NIZKs, instead of interacting with a verifier, the prover interacts with this random oracle. The prover generates its proof by querying the oracle with certain values and using the responses to craft a proof. This proof can then be verified by anyone, without further interaction, by also consulting the random oracle in a similar manner. The idea is that the hash function emulates the behavior of a random oracle, producing outputs that appear random and are difficult to predict. This ensures that a reasonable result can only be produced if the prover has the actual solution, and not with the incorrect solution.

It is important to note that the random oracle model is a theoretical construct, and its use is a topic of debate in the cryptographic community. While it allows for the design of elegant and efficient cryptographic schemes, like NIZKs, relying on the heuristic of treating hash functions as truly random oracles might introduce unforeseen vulnerabilities In summary, Non-Interactive Zero-Knowledge proofs in the random oracle model provide a way to achieve zero-knowledge properties without the need for interaction between the prover and verifier.

Resources Used:

Professor Heath's Lecture Slides

Boneh, D., Shoup, V. (2017). A graduate course in applied cryptography. Princeton University Press. (Chapter 20)