6

Zero Knowledge Proofs

Great Ideas in Theoretical Computer Science
Saarland University, Summer 2014

Proofs are the evidence of correctness of the assertions, and people can verify the correctness by reading the proof. However, we obtain much more than the correctness itself: After you read one proof of an assertion, you know not only the correctness, but also why it is correct. Is it possible to solely show the correctness of an assertion without revealing the knowledge of proofs? It turns out that it is indeed possible, and this is the topic of today's lecture: Zero Knowledge Systems.

The notion of Zero Knowledge Systems is introduced by Shafi Goldwasser, Silvio Micali, and Charles Rackoff in 1985 [4], and this notion has vast impact in Cryptography. Informally, Zero Knowledge provides an effective way of presenting the correctness of an assertion without revealing the actual proof.

6.1 One Motivating Example

We start with the following example from Wikipedia. Assume that Peggy has uncovered the secret word used to open a magic door in a cave. The cave is shaped like a circle, with the entrance on one side and the magic door blocking the opposite side. Victor wants to know whether Peggy knows the secret word; but Peggy, being a very private person, does not want to reveal the fact of her knowledge to the world in general.

They label the left and right paths from the entrance A and B. First, Victor waits outside the cave as Peggy goes in. Peggy takes either path A or B; Victor is not allowed to see which path she takes. Then, Victor enters the cave and shouts the name of the path he wants her to use to return, either A or B, chosen at random. Providing she really does know the magic word, this is easy: she opens the door, if necessary, and returns along the desired path.

However, suppose she did not know the word. Then, she would only be able to return by the named path if Victor were to give the name of the same path that she had entered by. Since Victor would choose A or B at random, she would have a 50% chance of guessing it correctly. If they were to repeat this trick many times, say 20 times in a row, her chance of successfully anticipating all of Victor's requests would become vanishingly small (about one in 1.05 million).

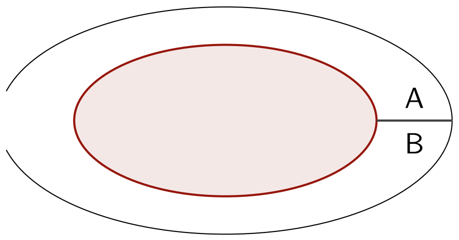

Figure 6.1: One motivating example of zero knowledge. Paths A and B are separated by a door, and Peggy needs to convince Victor that she has the key to open the door without showing the key to Victor.

Thus, if Peggy repeatedly appears at the exit Victor names, he can conclude that it is very probable that Peggy does in fact know the secret word. However, even if Victor is wearing a hidden camera that records the whole transaction, the only thing the camera will record is Victor shouting "A!" and Peggy appearing at A; Victor shouting "B!" and Peggy appearing at B. A recording of this type would be trivial for any two people to fake (requiring only that Peggy and Victor agree beforehand on the sequence of A's and B's that Victor will shout). Such a recording will certainly never be convincing to anyone but the original participants. In fact, even a person who was present as an observer at the original experiment would be unconvinced, since Victor and Peggy might have orchestrated the whole "experiment" from start to finish.

6.2 Zero Knowledge Proof

A zero knowledge proof is a game between a prover and a verifier. For a given assertion, the goal of a prover is, through the conversation with the verifier, to show the correctness of the assertion to the verifier without revealing the actual proof.

Definition 6.1 (Zero Knowledge Proofs)**.** *Let L be an NP-language, and let M be a polynomial time Turing machine such that $x \in L$ iff there is a $u \in \{0,1\}^{p(|x|)}$ such that $M(x,u) = 1$ where $p()$ is a polynomial.*

A pair P, V of interactive probabilistic polynomial-time algorithms is called a zero knowledge proof for L, if the following three conditions hold:

- *Completeness: For every $x \in L$ and u a certificate for this fact (i.e., $M(x,u) = 1$),*

$$\mathbf{Pr}\left[\,\mathsf{Out}_V\langle P(x,u), V(x)\rangle = 1\,\right] \geq 2/3,$$

where $\langle P(x,u), V(x)\rangle$ denotes the interaction of P and V where P gets x, u as input and V gets x as input, and $\mathsf{out}_V I$ denotes the output of V at the end of the interaction.

- *Soundness: If $x \notin L$, then for every strategy $P^\star$ and input u,*

$$\mathbf{Pr}\left[\,\mathsf{Out}_V\langle P^\star(x,u), V(x)\rangle = 1\,\right] \leq 1/3.$$

(The strategy $P^\star$ need not run in polynomial time.)

- *Perfect Zero Knowledge: For every probabilistic polynomial-time interactive strategy $V^\star$, there exists an expected probabilistic polynomial-time (stand-alone) algorithm $S^\star$ such that for every $x \in L$ and u a certificate for this fact, it holds that*

$$\mathsf{Out}\langle P(x,u), V^\star(x)\rangle \equiv S^\star(x).$$

(That is, these two random variables are identically distributed even though S does not have access to any certificate for x.) This algorithm $S^\star$ is called the simulator for $V^\star$, as it simulates the outcome of $V^\star$'s interaction with the prover.

Here the role of the prover is to convince the verifier in the fact that $x \in L$, but the verifier is interested in getting more information. The zero knowledge condition means the the verifier cannot learn anything new from the interaction, even if she does not follow the protocol but rather uses some other strategy $V^\star$. This fact is equivalent to say that she will learn the same thing by simply running the stand-alone algorithm $S^\star$ on input x. The perfect zero knowledge can be relaxed by conditioning on the probability distributions of $\mathsf{Out}\langle P(x,u), V^\star(x)$, and $S^\star(x)$. Basically, this condition states that whatever can be efficiently obtained by interacting with a prover, could also be computed without interaction, just by assuming that the assertion is true and conducting some efficient computation.

6.3 A ZKP for Graph Non-Isomorphism

We introduce a zero knowledge proof system for the graph non-isomorphism problem. We first recall the definition of graph isomorphism.

Definition 6.2 (graph isomorphism)**.** *Two graphs $G = (V_1, E_1)$ and $H = (V_2, E_2)$ with $|V_1| = |V_2| = n$ are called* isomorphic *if and only if there exists a permutation $\pi \in S_n$ such that $\{u, v\} \in E_1$ iff $\{\pi(u), \pi(v)\} \in E_2$.*

It is believed that this problem is not NP-complete. However any algorithm that runs faster than $O(2^{\sqrt{n}})$ is not known.

We first look at a zero-knowledge proof for graph non-isomorphism. We have two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ as an input of a game, and there are two players in the game exchanging information. The goal of one player, the prover, is to convince the other player, the verifier, that these two graphs G_1 and G_2 are not isomorphic. The zero knowledge protocol is shown in Algorithm 6.1.

Algorithm 6.1 A ZKP for Graph Non-Isomorphism

```
1: The verifier picks a random $b \in \{1,2\}$ and a permutation $\pi : V \mapsto V$ and sends $G = \pi(G_b)$
   to the prover.
2: The prover finds the bit $a \in \{1,2\}$ such that $G_a$ and $G$ are isomorphic and sends $a$ to the
   verifier.
3: The verifier checks that $a = b$, and, if so, accepts.
```

Theorem 6.3. *Let P be the prover algorithm and V be the verifier algorithm in the above protocol. Then*

1. *Completeness: If G_1 and G_2 are not isomorphic, then the interaction ends with the verifier accepting with probability 1.*
2. *Soundness: If G_1 and G_2 are isomorphic, then for every alternative prover strategy $P^\star$, of arbitrary complexity, the interaction ends with the verifier accepting with probability $1/2$.*

Proof. The first part of the theorem is true as for every permutation $\pi(G_1)$ is not isomorphic to G_2 and similarly for every permutation $\pi(G_2)$ is not isomorphic to G_1, therefore if G_1 and G_2 are not isomorphic, then no relabeling of G_1 can make it isomorphic to G_2. Since the prover is computational unbounded, he can always find out which graph the verifier has started from and, therefore, the prover always gives the right answer.

The second part of the theorem is true as there exists a permutation $\pi^\star$ such that $\pi^\star(G_2) = G_1$. Then if verifier picks a random permutation π_R then the distribution we obtain by $\pi_R(\pi^\star(G_2))$ and the distribution $\pi_R(G_1)$ are exactly the same as both are just random relabelling of, say, G_1. Therefore here the answer of the prover is independent on b and the prover succeeds with probability half. □

This probability of $1/2$ can be reduced to 2^{-k} by repeating the protocol k times. The reason why the verifier is convinced is that the prover would need to do something that is information theoretically impossible if the graphs are isomorphic. Therefore, it is not the answers themselves that convince the verifier but the fact that prover can give those answers without knowing the isomorphism.

6.4 A ZKP for Graph Isomorphism

Suppose now that the prover wants to prove that two given graphs G_1, G_2 are isomorphic, and that the prover knows an isomorphism. Algorithm 6.2 is a protocol for this problem. The input of the protocol are two graphs $G_1 = (V, E_1)$, $G_2 = (V, E_2)$. The prover knows the permutation $\pi^\star$ such that $\pi^\star(G_1) = G_2$. The prover wants to convince the verifier that the graphs are isomorphic without showing the permutation $\pi^\star$.

Algorithm 6.2 A ZKP for Graph Isomorphism

```
1: The prover picks a random permutation $\pi_R : V \to V$ and sends the graph $G \triangleq \pi_R(G_1)$
2: The verifier picks at random $b \in \{1, 2\}$ and sends $b$ to the prover.
3: The prover sends back $\pi_R$ if $b = 1$, and $\pi_R \circ (\pi^\star)^{-1}$ otherwise.
4: The verifier checks that the permutation $\pi$ received at the previous round is such that
   $\pi(G_b) = G$, and accepts if so.
```

Remark 6.4. *If the input graphs are isomorphic, then the graph send in step 1 is isomorphic to both input graphs. However, if the input graphs are not isomorphic, then no graph can be* *isomorphic to both graphs.*

Theorem 6.5. *Let P be the prover algorithm and V be the verifier algorithm in the above protocol. Then*

1. *Completeness: If G_1, G_2 are isomorphic, then the interaction ends with the verifier accepting with probability 1.*
2. *Soundness: If G_1, G_2 are not isomorphic, then for every alternative prover strategy P^* of arbitrary complexity, the interaction ends with the verifier accepting with probability $1/2$.*

Proof. By construction, the first statement of Theorem 6.5 holds. Now we see what happens if G_1 and G_2 are not isomorphic and the prover is not following the protocol and is trying to cheat a verifier. Since in the first round the prover sends a graph G, and G_1 and G_2 are not isomorphic, then G can not be isomorphic to both G_1 and G_2. So in second round with probability at least half the verifier is going to pick G_b that is not isomorphic to G. When this happens there is nothing that the prover can send in the third round to make the verifier accept, since the verifier accepts only if what prover sends in the third round is the isomorphism between G and G_b. Hence the prover will fail with probability a half at each round. Moreover, if we do the same protocol for several rounds the prover will be able to cheat only with exponentially small probability. □

6.5 ZKPs for NP-Complete Problems

We introduce the notion of *commitment schemes*. Commitment schemes are digital analogies of sealed envelopes (or, better, locked boxes). Sending a commitment means sending a string that binds the sender to a unique value without revealing this value to the receiver (as when getting a locked box). De-committing to the value means sending some auxiliary information that allows to read the uniquely committed value. Formally, a commitment scheme is an efficient two phase two-party protocol through which one party, called the sender, can commit itself to a value so the the following two conditions hold:

1. Secrecy: At the end of the first phase, the other party, called the receiver, does not gain knowledge of the sender's value. This requirement has to be satisfied for any polynomial time receiver.
2. Unambiguity: Given the transcript of the interaction in the first phase, there exists at most one value which the receiver may later (i.e., in the second phase) accept as legal "opening" of the commitment. This requirement has to be satisfied even if the sender tries to cheat (no matter what strategy it employs.)

In addition, we require that the protocol is viable in the sense that if both parties follow it then, at the end of the second phase, the receiver gets the value committed by the sender.

By assuming the existence of commitment schemes, there exists zero knowledge proofs of membership in any NP set. Let us look at the following problem for example.

Problem 6.6 (3-coloring problem). *Let $G = (V, E)$ be a graph with vertex set $V = \{1, \dots, n\}$. The 3-coloring problem asks if there is a coloring function $\phi : V \mapsto \{1, 2, 3\}$ such that $\phi(u) \neq \phi(v)$ for any edge $\{u, v\} \in E$.*

Algorithm 6.3 A ZKP for 3-Coloring Problem

```
1: The prover selects uniformly a permutation $\pi$ over $\{1,2,3\}$. For $i = 1$ to $n$, send the verifier a commitment to the value $\pi(\phi(i))$.
2: The verifier selects uniformly an edge $e \in E$ and send it to the prover.
3: Upon receiving $e = \{i, j\} \in E$, the prover decommits to the $i$th and $j$th values sent in the first step.
4: The verifier checks whether or not the decommitted values are different elements of $\{1,2,3\}$.
```

We can repeat the protocol $t \cdot |E|$ times so that the soundness error probability is bounded by $\exp(-t)$. Every time that we repeat it, a prover need to select a new permutation π, so crucially the colors that the verifier sees in one round have nothing to do with the color that he saw from previous rounds (noticing that, if the prover always use the same colors, then the verifier would know the original colors by just asking sufficient number of times).

By using the standard Karp-reduction to 3-Colorability, the protocol above can be used for constructing zero knowledge proofs for any problem in NP.

6.6 Brief History and Reference

The concept of zero-knowledge was introduced by Goldwasser, Micali and Rackoff [4]. The early version of their paper has existed as early as in 1982, and were rejected three times from major conferences (FOCS '83, STOC '84, FOCS '84) before appearing in STOC '85. Goldreich, Micali and Wigderson [3] showed how to construct zero-knowledge systems from any NP-set.

Goldreich's article *A short Tutorial of Zero-Knowledge* is an excellent reference for studying zero-knowledge. The earliest version of the article appeared in 2002 titled *Zero-Knowledge twenty years after its invention* [2]. In addition, Aaronson [1] discusses zero knowledge proofs from a Philosophical point of view.

References

[1] Scott Aaronson. Why philosophers should care about computational complexity. *CoRR*, abs/1108.1791, 2011.

[2] Oded Goldreich. Zero-knowledge twenty years after its invention. *Electronic Colloquium on Computational Complexity (ECCC)*, (063), 2002.

[3] 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*, 38, 1 1991.

[4] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing (STOC'85)*, pages 291–304, 1985.