UCSB CS 291D: Blockchains and Cryptocurrencies **Fall 2020**

Lecture 8: Intro to Zero Knowledge Proofs

Lecturer: Shumo Chu *Scribes: Xinlu Zhang, Zach Sisco*

8.1 Proof, Secrets, and Computation

8.1.1 What are proofs?

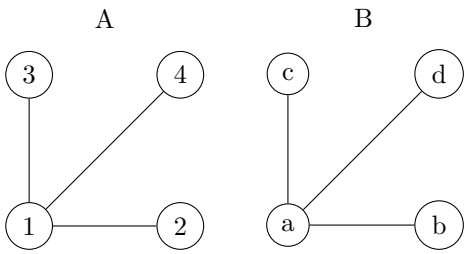
Goedel and Turing claimed proofs are strings with special syntactic properties. Using the conventional notion of a mathematical proof, to prove that a statement is true, one should provide a sequence of symbols that can be written down and a valid sequence exists only for true statements.

In computational complexity theory, NP (nondeterministic polynomial time) is is the class of languages in which membership can be verified quickly: for all $x \in L$, this fact can be verified in polynomial time.

Example: Graph Isomorphism

We say two graphs $H$and G are isomorphic if they are the same up to a renumbering of vertices. In other words, if there is a permutation π of the notations of the nodes of G such that $\pi(G) = H$.
Mathematically, we have two graphs are isomorphiic s.t.

$$ISO \equiv \{(G, H) : G \cong H\}$$



For example, shown above, graphs A and B are isomorphic because such a permutation $\pi(A)$ exists such that two vertices are connected by an edge in A if and only if there are a corresponding pair of vertices that are connected by an edge in B. This permutation would map the following vertices: 1 to a; 2 to b; 3 to c; and 4 to d. The edges are mapped as follows: the $(1, 2)$ edge maps to the (a, b) edge; the $(1, 3)$ edge maps to the (a, c) edge; and the $(1, 4)$ edge maps to the (a, d) edge.

■

8.1.2 Interactive Proofs

The interactive proof is a more general way for people to convince one another of the validity of a statement. There are two parties involved:

- Prover: has infinite running time - like a wizard.
- Verifier: has polynomial running time, and has the ability to generate random numbers. This is known as probabilistic polynomial time (PPT) — can think of as a king who asks questions.

The prover and verifier interact for a polynomial time. Finally, the verifier accepts $x \in L$ or rejects since it is not convinced that $x \in L$. (x may be in L, but rejection means that the prover did not convince the verifier.)

Informally, there are two objectives of a proof systems:

- Completeness: It's possible to prove all true theorems.
- Soundness: It's not possible to prove a false theorem.

We say that if a prover and verifier forms a system such that:

- There is a small probability of completeness error;
- There is a small probability of soundness error;

The system is an Interactive Proof System for membership in L.

Definition 8.1 ***(Interactive Proof)*** *(P,V) is an Interactive Proof System (IPS) for a language L with a security parameter k, which the following two properties hold:*

- ***Completeness****:* $\forall x \in L,\ Prob[(P,V)[x] = accept] \geq 1 - negl(k)$
- ***Soundness****:* $\forall x \notin L,\ Prob[(P^\star,V)[x] = accept] \leq negl(k)$

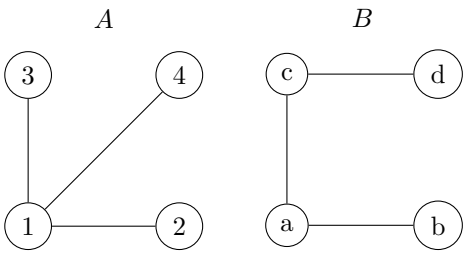
where negl is a negligible function and k is used to control the error probability [CS282].

In the above definition, the P,V means honest algorithm but $P^\star$ denotes possibility malicious prover.

Example: Graph Non-isomorphism:

We say that a pair of graphs G, H are not isomorphic s.t. $NISO \equiv \{(G,H) : G \ncong H\}$

To prove this in an IPS, we set the proof up as a game between a prover and a verifier with the following conditions: if the theorem is true, the prover always wins; if the theorem is false, the prover loses to the verifier at least half of the time.



Consider the graphs, A and B, above. To show $A \ncong B$, the game plays out as follows. The verifier makes a permutation of A, $\pi(A)$, and sends the corresponding graph $C = \pi(A)$ to the prover, asking which graph

C is isomorphic to. But by construction, $C \cong A$, and this is the answer the prover gives. In this way, the prover will always win at this game no matter how many times it is played.

On the other hand, knowing that false theorems are not provable in an IPS, consider the game that attempts to show $A \cong B$. Again, the verifier makes a permutation of A, $\pi(A)$, and sends the corresponding graph $C = \pi(A)$ to the prover, asking which graph C is isomorphic to. However, C is both isomorphic to A and B. The prover does not know this, so there is a $1/2$ probability of the prover giving the correct answer. As the game repeats, this probability drops exponentially, showing that the prover will lose this game at least half of the time.

■

Formally, the process that describes the IPS for proving graph non-isomorphism, for some graphs G_1 and G_2, is as follows [COS522].

1. V : pick a random number $i \in \{1, 2\}$ uniformly. Randomly permute the vertices of G_i to get a new graph C. Send C to P.
2. P: identify which of G_1, G_2 was used to produce C. Let G_j be that graph. Send j to V.
3. V : accept if $i = j$; reject otherwise.

If G_1 and G_2 is not isomorphic, then there is a prover s.t. $Pr[Vaccepts] = 1$, because a pover can tell which graph is isomorphic to C, if these two graphs are non-isomorphic. However, G_1 and G_2 is isomorphic, the best the prover can do is uniformly randomly guess a result, s.t. $Pr[Vaccepts] \leq 1/2$.

8.1.3 Zero-knowledge Proofs

Intuitively, a zero-knowledge proof is a method in which the prover can prove to the verifier that a given statement X is true without revealing additional information. A result of this kind of proof system is that everything provable is in fact provable in zero knowledge.

The base axiom of a zero-knowledge proof system is that polynomial-time computations and random bit strings are free. From this basis, zero knowledge proofs are constructed.

Example: Graph isomorphism, $A \cong B$

As the verifier (without the prover), generate a graph A—this can be easily done in polynomial time and follows from the axiom that efficient computation is free. Then, generate a random renaming of the veritices in A to obtain a new correspondence $\phi(A) = C$. By construction, $A \cong C$, and thus $A \cong B$ because we now have a valid correspondence $\phi(A)$.

■

The key idea with the previous example is giving a proof that $A \cong B$ without revealing any additional knowledge.

For reference, a formal description of a zero-knowledge proof system from [CS282] follows.

Definition 8.2 . *(Zero-Knowledge Proof). An Interactive Proof $PV^\star$ for a language L is said to be Zero-Knowledge if*

$$\forall V^\star \in PPT \quad \exists S_{v^\star} \in PPT \quad \forall x \in L \quad s.t.$$

$$[P \leftrightarrow V * [x]] \cong [S_{V^\star}(x)]$$

For every verifier $V^\star$, there is an expected polynomial-time algorithm simulator $S_{V^\star}$, which can produce transcripts, without interacting with the real prover, based on the input of assertion x which needs to be proven. The produced transcripts are indistinguishable from interaction results with the real prover or more specifically points from $[PV^\star(x)]$.

Example: Passwords

Zero-knowledge proofs can be used for authentication. For instance, a user (the prover) can associate some large number with their account on a web service (the verifier). If the user can provide proof that the number is a product of two primes then they obtain access to the account. This works as a zero-knowledge proof because the verifier gains no additional knowledge from the proof; it only knows the number associated with the user. The user merely provides proof of the prime factorization for that number.

Zero-knowledge proof passwords are a stronger alternative to conventional passwords because the actual secret is not stored on the verifier's end.

■

8.1.4 Other Proofs

8.1.4.1 Samplable Proofs

Proofs can be long, and in order to verify a proof we must read every line. However, the intuition with samplable proofs is to sample parts of the proof and check that their relationship holds. If so, we conclude the proof is correct. By repeating the sampling process we gain confidence of the proof's correctness, as the probability of error decreases exponentially each time. This technique is somewhat inefficient because the proof must be massaged into a special format that may increase the length of the proof. A formal description of a sampling proof system follows.

Formally, a sampling proof system is based on the interactive proof-system, (SP, SV) [Micali]. The sampling prover SP computes a samplable proof $x \in L$, string W_x, based on an input x (a n-bit long string belonging to a given NP-language L) and w_x (a NP-witness that indeed $x \in L$). The sampling verifier SV outputs either 'ACCEPT' or 'REJECT' based on input string x (candidate member of L) and random access to σ (candidate samplable proof for $x \in L$) by reaching polynomial-log(n) bit-locations of σ. SV outputs ACCEPT if $x \in L$ and $\sigma = SP(x, w_x)$ for some correct w_x. However, if $x \notin L$, $\forall\sigma$ SV outputs REJECT with probability $\frac{1}{2}$.

8.1.4.2 Computationally Sound (CS) Proofs

The most important property of CS proofs is that efficient proofs exist for all theorems. A CS proof system is both complete and inconsistent. That is, we can say, "X and not-X are always provable," but proving a false theorem is extremely hard. The advantage of this is that a proof for something true is easy to find.

8.1.4.3 Rational Proofs

Rational proofs exist at the intersection of mathematics, computer science, and economics. The idea is to "pay for a proof." So obtaining the truth becomes the maximization of computable functions. For instance, if Alice wants to know what $f(x)$ is, she can pay Bob. Bob says $f(x)$ is 27. Alice knows that Bob is telling

the truth because the price she paid is a function of f, x, and 27. This works because Bob wants to maximize the payment he receives for telling the truth, $f(x)$.

References

[COS522] "Chapter 9: Interactive proofs". [Online]. Accessed on: November 5, 2020. Available: https://www.cs.princeton.edu/courses/archive/spring06/cos522/ip.pdf

[CS282] "Foundations of Cryptography: Lecture 9". [Online]. Accessed on: November 5, 2020. Available: http://web.cs.ucla.edu/~rafail/TEACHING/WINTER-2005/L9/L9.pdf

[Micali] "Computationally-Sound Proofs". Micali, S. Logic Colloquium '95: Proceedings of the Annual European Summer Meeting of the Association of Symbolic Logic, held in Haifa, Israel, August 9-18, 1995, 214–268, Springer-Verlag, Berlin, 1998. Available: https://projecteuclid.org/euclid.lnl/1235415908