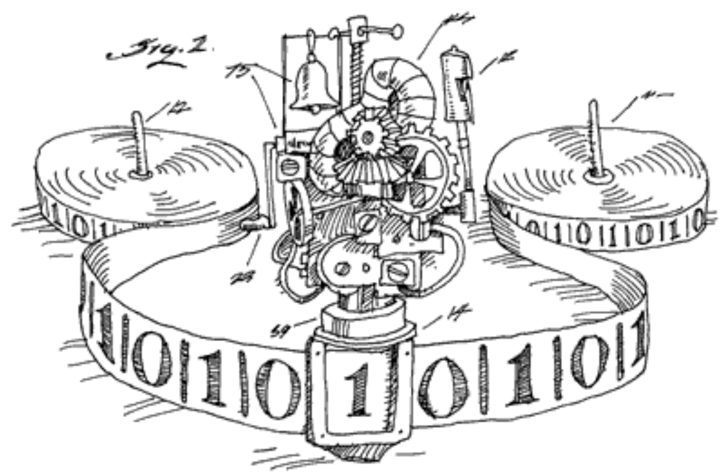
EECS 376: Foundations of Computer Science

Lecture 25 – Interactive Proofs & Zero-Knowledge proofs



Seminal Papers

- S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, STOC '85: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, December 1985.
- S. Goldwasser, S. Micali, A. Wigderson, Proofs that yield nothing but their validity and a methodology of cryptographic protocol design, 27th Annual Symposium on Foundations of Computer Science, October 1986.

co-originators of the important concepts of "interactive proofs" and "zero-knowledge proofs"

Goldwasser and Micali win Turing Award

Team honored for 'revolutionizing the science of cryptography.'

Abby Abazorius, CSAIL
March 13, 2013

Including introducing interactive proof systems and zero-knowledge proofs

Chris Peikert's PhD advisor

Shafi Goldwasser Silvio Micali

co-originators of the important concepts of "interactive proofs" and "zero-knowledge proofs"

C.Rackoff

Co-won the 1993 Godel Prize

A. Wigderson

Won the Turing Award last year!

For “reshaping our understanding of the role of randomness in computation”

Interactive Proof System

An interactive proof system is a concept in computational complexity theory that models computation as a dialogue between two parties: a prover and a verifier.

- Prover: This party has unlimited computational resources but cannot be trusted.
- Verifier: This party has limited computational power but is assumed to be always honest.

Interactive Proof System

- The prover and verifier exchange messages to determine whether a given string belongs to a language.
- The interaction continues until the verifier is convinced of the answer to the problem.

Properties of Interactive Proof Systems

Two main properties characterize interactive proof systems:

- Completeness: If the statement is true, an honest prover can convince the honest verifier of its truth.
- Soundness: If the statement is false, no prover can convince the honest verifier that it is true, except with some small probability.

Applications of Interactive Proof Systems

Interactive proof systems have a wide range of applications across various fields. These applications demonstrate their versatility in enhancing security, verifying correctness, and solving complex computational problems.

- Cryptographic Protocols: They are fundamental in designing protocols where trust and verification are crucial, such as in zero-knowledge.
- Approximation Algorithms: Interactive proofs help understand the power and limitations of approximation algorithms.

Applications of Interactive Proof Systems

- Program Checking: They are used to check the correctness of programs in a way that the verifier can be convinced of the program's correctness without having to execute the program.
- Complexity Theory: Interactive proofs have provided insights into the hardness of certain computational problems, such as graph isomorphism and the approximate shortest lattice vector problem.
- Interactive Theorem Provers: These are tools used to certify mathematical theories and construct machine-verified proofs.

Interactive Proof System

(with k-round interactions)

- Let L $\subseteq \{0.1\}^*$ be a language and (A, B) an interactive pair of Turing machines. We say that (A, B) is an *interactive proof system* for L if A (the prover) has infinite power, B (the *verifier*) is polynomial time and they satisfy the following properties.
  - For any x ∈ L given as input to (A, B), B halts and accepts with probability 1-$\frac{1}{n^k}$ for each k and sufficiently large n.
  - For any ITM A* and any x not in L given as input to (A*,B), B accepts with probability at most $\frac{1}{n^k}$ for each k and sufficiently large n.

Interactive Complexity Classes

- We define **IP**, *Interactive Polynomial-time*, to be the class of languages possessing an interactive proof system.
- In this case, we may also say that L is interactively provable. To emphasize that the prover has unlimited power, we may write $\mathbf{IP}_\infty$ for **IP**.

Graph Isomorphism

- We say two graphs G1 and G2 are isomorphic if they are the same up to a renumbering of vertices. In other words, if there is a permutation π of the labels of the nodes of G1 such that $\pi(G1) = G2$
- The graph isomorphism problem is important in a variety of fields and has a rich history.
- Along with the factoring problem, it is the most famous NP problem that is not known to be either in P or NP-complete.

Protocol: Private-coin Graph (Non-)isomorphism

- Both graph isomorphism and its complement can be shown to belong to IP as follows.

- The protocol has k - round interactions. Each round proceeds as follows:

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

Interactive Complexity Classes

- If we allow the probabilistic verifier machine and the all-powerful prover to interact for a polynomial number of rounds, we get the class of problems called **IP**.
- In 1992, Adi Shamir was able to prove one of the central results of complexity theory that **IP** equals **PSPACE**, the class of problems solvable by an ordinary deterministic Turing machine in polynomial space.

Zero Knowledge

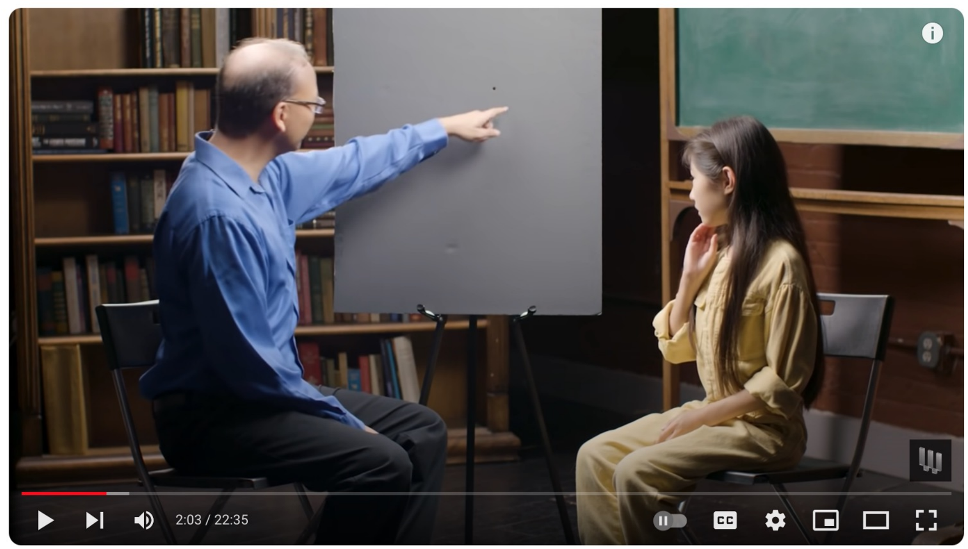
What is a Zero-Knowledge Proof?

A convincing demonstration that some statement is true—
without revealing anything beyond the fact that it's true!

Example: Suppose you knew that these two pens are different. How could you convince me they're different without telling me what is different about them?

Zero-Knowledge Proof for “Where’s Waldo?”
(or “where’s the puffin among the penguins?”)

How could you convince me that there is a puffin in this picture without revealing its location (or anything else)?



Computer Scientist Explains One Concept in 5 Levels of Difficulty | WIRED

How are Zero-Knowledge Proofs Useful?

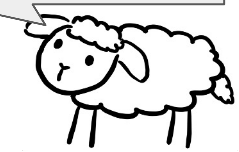
- **Electronic money / Blockchain:** Prove that your account has enough money for a transaction, without revealing your balance.

- **Group signatures:** Prove that a member of a certain group signed a message, without revealing which one did.
  - E.g., give students keycard access to restricted areas without tracking individual students.

- **Multi-party computation:** Compute aggregate functions of private data (e.g., medical records), without revealing individual data.

The Model

Both parties know x and L



Let's put aside zero knowledge for a moment and recall **verifying a certificate for an instance x of a language L in NP.**

Given an instance x (of an NP-language L), I send a certificate **c**

c

I run a verification algorithm V(x,**c**)

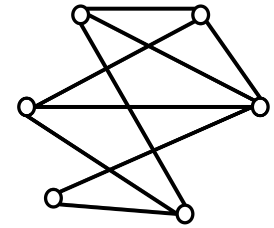
Merlin
"the prover"
All-powerful but untrustworthy

Recall that a language L is in NP if:

- **x ∈ L** ⇒ Merlin can give Arthur a convincing "proof" that x ∈ L, i.e., *exists* certificate c so that V(x,c) accepts
- **x ∉ L** ⇒ Merlin can't "fool" Arthur into accepting that x ∈ L, i.e., *for any* c, V(x,c) rejects.

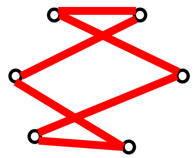
Arthur
"the verifier"
Restricted to poly-time computation

Hamiltonian Cycle



G

I send a Hamiltonian Cycle in G



I check it, and am convinced that G has a Hamiltonian Cycle

This is NOT a **ZK** proof because Arthur learns an actual Ham Cycle (not just the fact that G has a Ham Cycle)!

Merlin
"the prover"
All-powerful but untrustworthy

Can we make this into a ZK proof?

YES!
By using *randomness* and *interaction*.

Arthur
"the verifier"
Restricted to poly-time computation

*Actually it's possible to make it non-interactive (we won't show)

Requirements for a protocol to be a ZK proof

1. **"Completeness": If $x \in L$,** Merlin can cause Arthur to accept.

2. **"Soundness": If $x \notin L$,** then Merlin cannot "fool" Arthur into accepting, <u>except</u> with some *small probability*. (← new, needed weakening)

3. **"Efficiency":** Arthur runs in polynomial time (in the size of x).

4. **"Zero knowledge":** Arthur "learns nothing" but the fact that $x \in L$.

Q: ***But what exactly does that mean? How to define it?***

A: Given that $x \in L$, whatever Arthur saw from Merlin could have been **generated by Arthur *on his own, without ever interacting with Merlin*.**

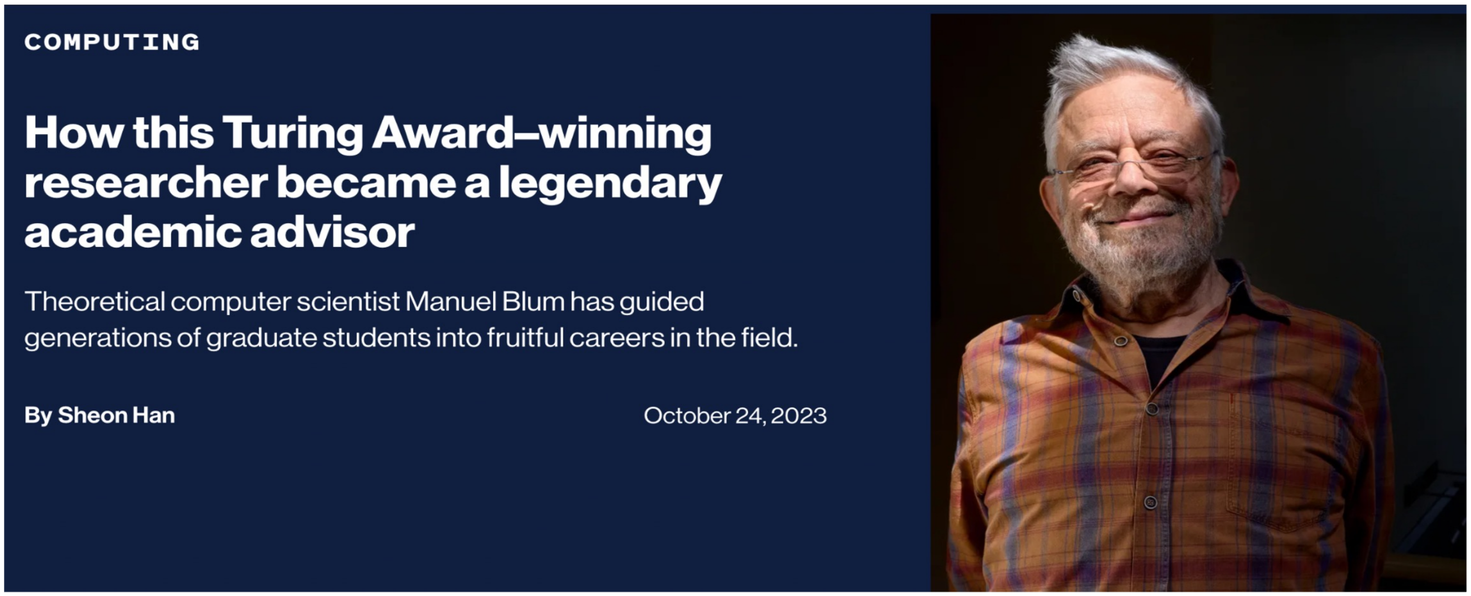
↳ If Merlin uses randomness, Arthur could have generated messages from the *same probability distribution* as Merlin's messages.

A Zero-Knowledge Proof for Hamiltonian Cycle

Merlin convinces Arthur that G has a Ham Cycle without revealing anything else

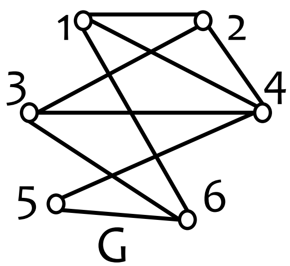
A Zero-Knowledge Proof for HamCycle [Blum ‘86]

Manuel Blum, How to Prove a Theorem So No One Else Can Claim It, Proceedings of the International Congress of Mathematicians, Berkeley, California, USA, 1986.



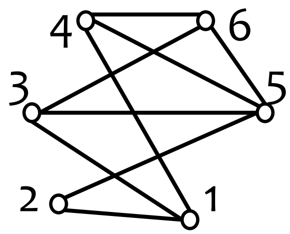
Some academic descendants of Blum in our department:
Chris Peikert, Euiwoong Lee, Ben Fish, Wei Hu

The Protocol has k - round interactions. Each round proceeds as follows



⇒

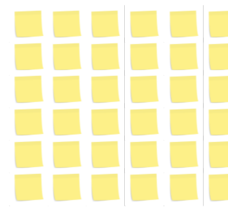
permutation: 4, 6, 3, 5, 2, 1

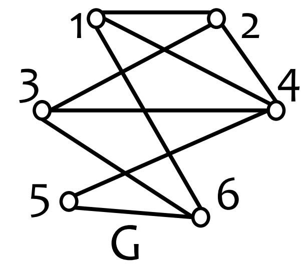


⇒

	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	1
4	1	0	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0

⇘





I randomly choose one *challenge*:

- **"reveal all"** or
- **"reveal cycle"**

I am supposed to:

- Randomly shuffle vertices.
- Write adjacency matrix.
- Cover each entry with a post-it note, and send!

"reveal all"

If **"reveal all"**, I reveal the *entire* adjacency matrix *and* the shuffling permutation

	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	1
4	1	0	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0

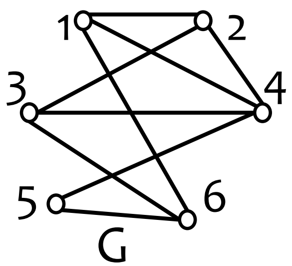
permutation: 4, 6, 3, 5, 2, 1

Accept if this matches original graph G

The Protocol has k - round interactions. Each round proceeds as follows

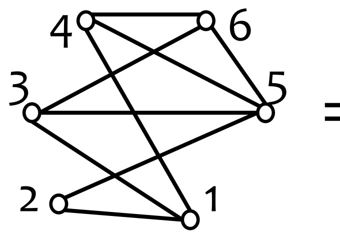
Quiz:

1. How would Merlin fool Arthur if **"reveal cycle" ?**
2. How about **"reveal all"** ?
3. What if Merlin didn't shuffle vertices ?



⇒

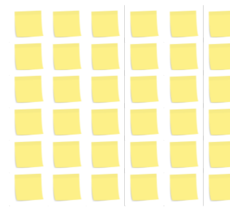
permutation: 4, 6, 3, 5, 2, 1



⇒

	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	1
4	1	0	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0

⇘

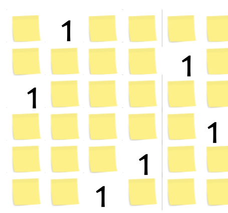


I am supposed to:
- Randomly shuffle vertices.
- Write adjacency matrix.
- Cover each entry with a post-it note, and send!

I randomly choose one *challenge*:
- **"reveal all"** or
- **"reveal cycle"**

"reveal cycle"

If **"reveal cycle"**, I reveal only the (shuffled) Ham Cycle

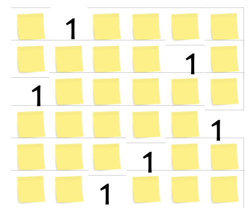


Accept if revealed entries are 1s, and correspond to a Ham Cycle (ignoring G itself)

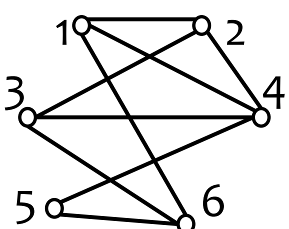
Analysis

"Efficiency": Arthur runs in polynomial time

- If **"reveal cycle"**, check in poly-time that the revealed part is a HamCycle



- If **"reveal all"**, check in poly-time that the initial graph and the revealed graph are the same using the permutation

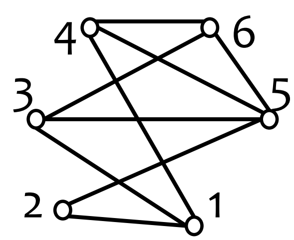


G

Arthur had this from the beginning

permutation: 4, 6, 3, 5, 2, 1

	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	1
4	1	0	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0



Arthur is then given this

Just relabel and check

Analysis

"Completeness":

If G has a Ham Cycle, Merlin can cause Arthur to accept:

Merlin follows the protocol.

If **"reveal cycle"**, Arthur can check the existence of HamCycle and accept.

If **"reveal all"**, Arthur can check that the two graphs are the same and accept.

"Soundness":

If G has **no** Ham Cycle, Merlin "fools" Arthur into accepting with prob ≤ ½:

If Merlin **modified** the graph, Merlin fails if **"reveal all"**.

If Merlin **did not modify** the graph, Merlin fails if **"reveal cycle"**

In both cases, Arthur is fooled with prob $\leq$ ½

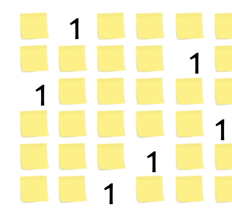
How can we reduce this "fooling" probability to near 0?

Analysis

“Zero knowledge”:
If G has a Ham Cycle,
then Arthur, ***without ever interacting with Merlin,*** could have generated messages from the *same probability distribution* as Merlin’s.

If **“reveal cycle”:** Arthur sees a HamCycle with randomly permuted vertices.

- Can he generate this on his own?
- Yes!
- **Note:** Arthur does not need to find HamCycle in the original graph

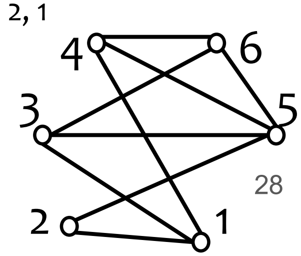


If **“reveal all”**, Arthur sees a random permutation and the original graph after vertex relabeling

- Can he generate this on his own?
- Yes!

permutation: 4, 6, 3, 5, 2, 1

	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	1
4	1	0	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0



Analysis

- Can you implement this protocol on the internet?
  - Merlin and Arthur are far away from each other.
  - What is missing?

- Not clear how to implement the digital version of "post-it"!

Digital Commitment Schemes

A digital version of this ZK proof requires a **"commitment scheme"**.

I.e., Merlin "commits" to bits without revealing them to Arthur, and can later reveal any of them as needed–without being able to *change* them.

This is possible!

(under cryptographic assumptions)

We won't prove it, but here's some intuition:

- Merlin "commits" to primes p,q by sending Arthur their product n=pq.
- Arthur can't factor efficiently, so doesn't know p and q.
- Merlin can later reveal p,q. Arthur checks that they are prime and that their product is n. No other numbers satisfy these properties, so Merlin could not have changed them!

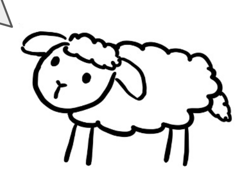
(Needs to be extended to allow Merlin to commit to a desired *bit*.)

Zero-Knowledge Proofs for NP (and Beyond)

There is a ZK proof for ANY problem in NP!

(under cryptographic assumptions)

This includes NP-complete problems, problems in P, decision versions of discrete log and factoring, …



How?

Let L be a language in NP.
We want a ZK proof that $x \in L$.

Let f be a p.t.m.reduction from L to HC
(exists since HC is NP-complete).

So, $x \in L \Leftrightarrow f(x) \in HC$.

To ZK-prove that $x \in L$, use the ZK proof for HC to prove that $f(x) \in HC$.

First ZK proof for an NP-C problem (with larger error) [Goldwasser-Micali-Wigderson '86]

[Goldwasser-Micali-Rackoff '85]

Beyond NP

Theorem (we won't show): Under cryptographic assumptions (in particular, the existence of OWFs), the set of problems with a **ZK proof** is *equivalent* to those in **PSPACE,** i.e., solvable w/ **polynomial space**.
[Lund-Karloff-Fortnow-Nisan-Shamir'92]

It is known that **NP** $\subseteq$ **PSPACE**, and it is conjectured that **NP** $\neq$ **PSPACE.**

An Unconditional Zero-Knowledge Proof

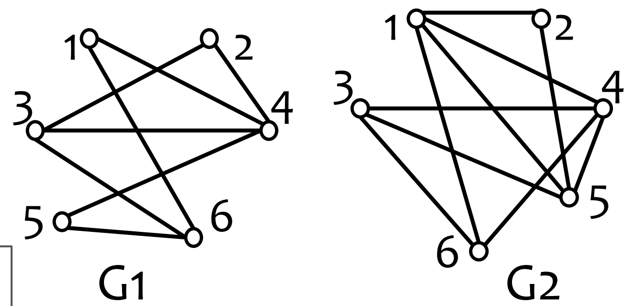
Unconditional ZK-Proof

Two graphs are *isomorphic* if one can be obtained from the other by relabeling the vertices.

Problem: Graph Non-Isomorphism
Given two graphs, are they "different" (non-isomorphic)?

Theorem:
There is a ZK proof for **Graph Non-Isomorphism** [GMW '86]

A ZK Proof for Graph Non-Isomorphism
The Protocol has k rounds. Each round proceeds as follows



I randomly choose

1. G1 or G2,
2. randomly relabel the vertices
3. send the resulting adjacency matrix!

I say whether it's a relabeled of G1 or G2

	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	1
4	1	0	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0

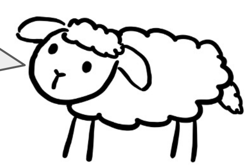
(a scrambling of G1, generated by Arthur)

It's G1!

I accept if Merlin got the right answer

Quiz: What if Arthur did not randomly relabel vertices? How would Merlin fool him?

This is reminiscent of the pen example



Analysis

Merlin can convince Arthur that G1 and G2 are non-isomorphic without revealing anything else.

"Completeness":

If G1 and G2 are **non-isomorphic**, Merlin can cause Arthur to accept:

Merlin can tell which relabeled graph he got,

Arthur see a correct answer.

"Soundness":

If G1 and G2 are **isomorphic**, Merlin "fools" Arthur into accepting with prob ≤ ½:

Merlin can't tell which scrambled graph he got,

Merlin's answer is correct with probability ½.

How can we reduce this "fooling" probability to near 0?

Analysis

Merlin can convince Arthur that G1 and G2 are non-isomorphic without revealing anything else.

"Zero knowledge":

If G1 and G2 are non-isomorphic, then

whatever Arthur saw from Merlin could have been

generated by Arthur *on his own, without ever interacting with Merlin*.

Arthur already knows the correct graph.

He can generate the correct answer.

"Efficiency": Arthur runs in polynomial time

Trivial.

Wrap Up

Reflection on **"Zero knowledge"**

Informal:

Arthur "learns nothing" but the fact that $x \in L$.

Formal:

Given that $x \in L$,
whatever Arthur saw from Merlin could have been
generated by Arthur *on his own,*
***without ever interacting with Merlin*.**

We can formalize vague concepts, like knowledge,
via **computational power**.
This is a deep insight from theoretical CS.

More example: What is **pseudo-random**? What is **private**? What is **secure**?
See [Imitation Games - Avi Wigderson]

Randomized Verifier is Essential

Claim: If a language L has a **ZK proof** where the verifier is **deterministic**, then L is in **P**.

Bottom line: if the verifier isn't randomized, then the prover cannot prove anything *in a zero-knowledge manner* than what the verifier can already solve on his own. So, it is not useful at all.

Randomized Verifier is Essential

Claim: If a language L has a **ZK proof** where the verifier is **deterministic**, then L is in **P**.

Without loss of generality, we can assume this:

1. There is only a single one-way message from Merlin.
   - Merlin knows Arthur's (deterministic) responses anyway.
   - So just concatenate all of Merlin's messages into one.

2. The strong **"Soundness":**
   If $x \notin L$, then Merlin "fool" Arthur **with probability 0.**
   - Otherwise, if fooling prob >0, there is a way to fool Arthur prob 1 (as Arthur is deterministic)

3. Merlin is deterministic.

So, w.l.o.g., Arthur is exactly the Verifier in the setting of NP problem

Randomized Verifier is Essential

Claim: If a language L has a **ZK proof** where the verifier is **deterministic**, then L is in **P**.

Since Arthur is exactly the Verifier in the setting of NP problem

- If x in in L, Merlin can send a message C and convince Arthur to accept.
- If x is not in L, for any Merlin's message C, Arthur always rejects.
- By the zero-knowledge condition, Arthur can come up with C in poly time without Merlin's help.
- So, Arthur can solve the problem in polynomial time on his own! L is in **P**.

Admin

Exam Review Session led by Erik:
Tuesday, June 25th, 1-3 pm, BBB 1670.

HW5 is due today at 8 pm.

Reminder: Filling out the course evaluations is 1% of your grade, which is otherwise covered by the final exam. Remember to submit the receipt on Gradescope.