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

Goldwasser and Micali win Turing Award

Team honored for 'revolutionizing the science of cryptography.'

Including introducing zero-knowledge proofs

Abby Abazorius, CSAIL
March 13, 2013

Chris Peikert's PhD advisor

Shafi Goldwasser

Silvio Micali

How are Zero-Knowledge Proofs Useful?

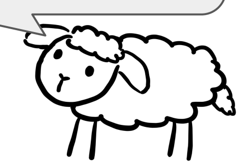
- **Electronic money / Blockchain:** Prove that your account has enough money for a transaction, without revealing your balance.

- **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**

Merlin
"the prover"

All-powerful but untrustworthy

c

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

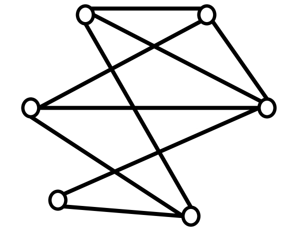
Arthur
"the verifier"

Restricted to poly-time computation

Recall that a language L is in NP if:

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

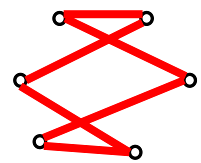
Hamiltonian Cycle



G

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

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 (rather than 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

- **"Completeness"**: **If x ∈ L,** Merlin can cause Arthur to accept.

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

- **"Zero knowledge"**: Arthur "learns nothing" but the fact that x ∈ L.

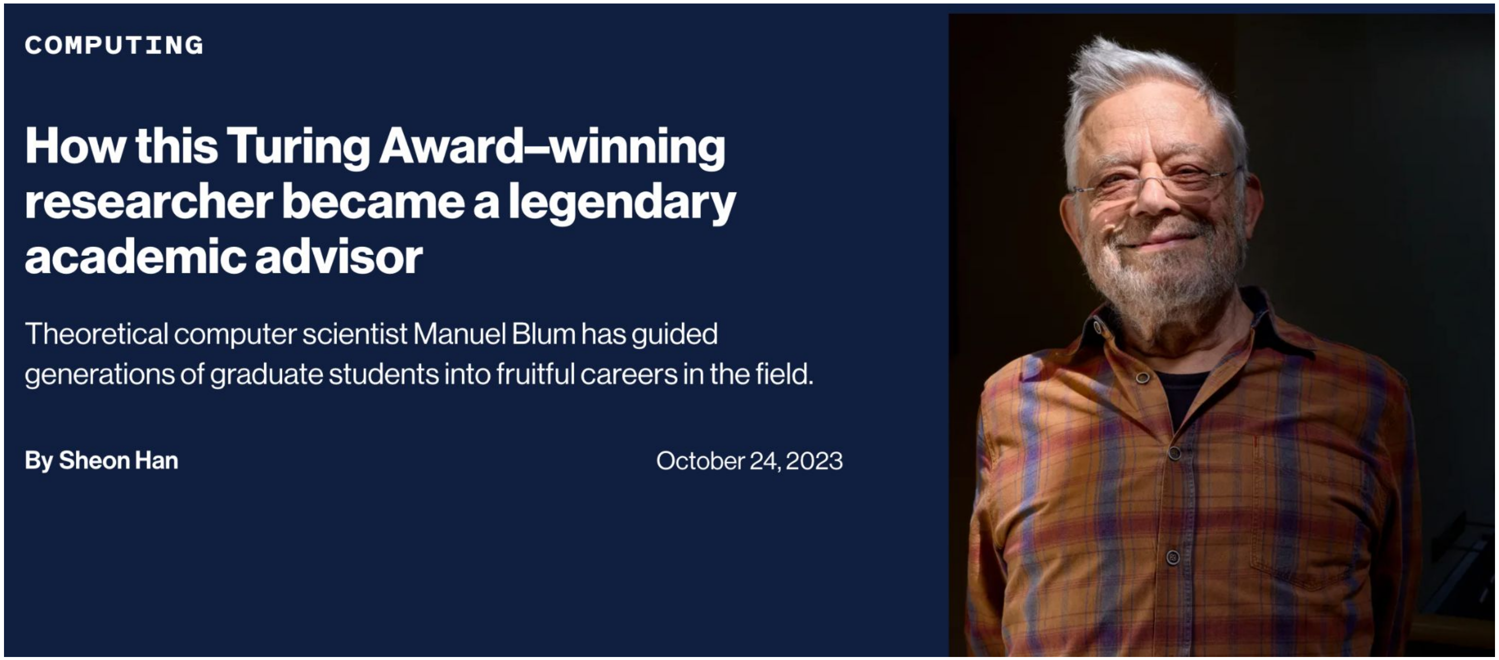
  Q: ***But what exactly does that mean? How to define it?***

  A: Given that x ∈ 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.

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

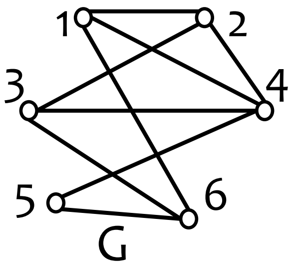
A Zero-Knowledge Proof for HamCycle [Blum ‘86]



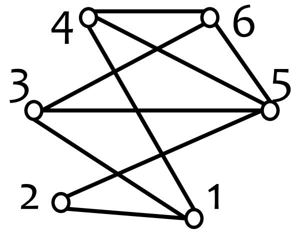
Some academic descendants of Blum in our department:
Chris Peikert, Euiwoong Lee, Ben Fish, Wei Hu

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

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



⇒



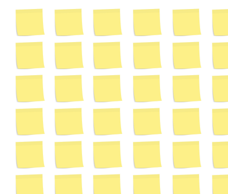
⇒

	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 all"

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

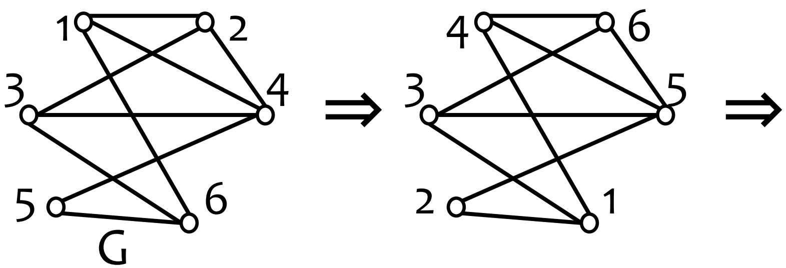
	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	0	1	1
4	1	0	0	0	1	1
5	0	1	1	1	0	1
6	0	0	1	1	1	0

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

Accept if this matches original graph G

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

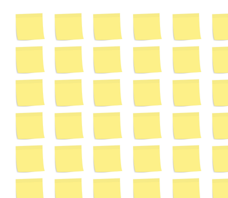
Merlin convincing Arthur that G has a Ham Cycle without revealing anything else



	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

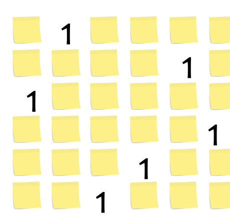
⇓

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)

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

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

- **"Completeness"**: If G has a Ham Cycle, Merlin can cause Arthur to accept:

- **"Soundness"**: If G has **no** Ham Cycle, then Merlin "fools" Arthur into accepting with probability ≤ ½:

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

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

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

- **"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.

- **"Efficiency"**: Arthur runs in polynomial time

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)

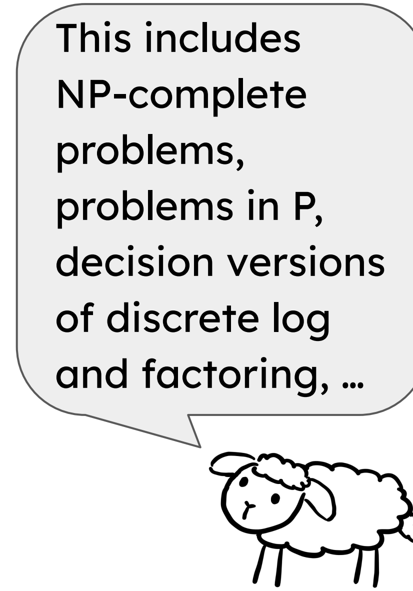
We won’t prove it, but here’s some intuition:

- 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*.)

This implies a ZK proof for ANY problem in NP!

(under cryptographic assumptions)



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) [Goldreich-Micali-Wigderson '86]

Won the Turing Award last week!
For "reshaping our understanding of the role of randomness in computation"

Beyond NP

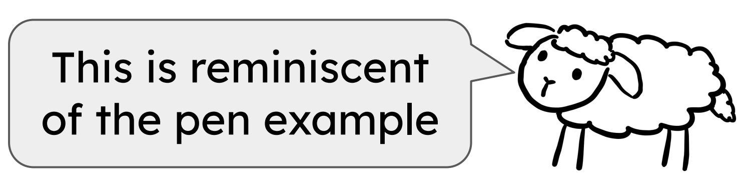
Theorem (we won't show): Under cryptographic assumptions, the set of problems that have a **ZK proof** is *equivalent* to the set of problems in **PSPACE,** i.e., solvable w/ **polynomial space.** [Lund-Karloff-Fortnow-Nisan-Shamir'92]

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

There are also problems that seem to be outside of NP that have ZK proofs, ***unconditionally***!

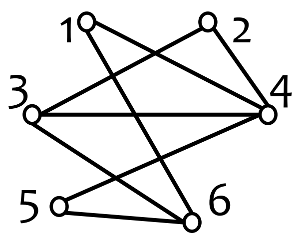
Example: There is a ZK proof for **Graph Non-Isomorphism:**
Given two graphs, are they "different" (non-isomorphic)? [GMW '86]

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

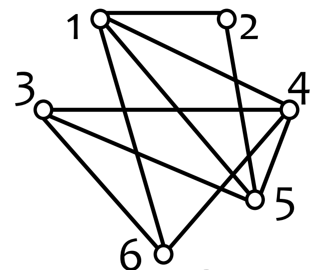


A ZK Proof for Graph Non-Isomorphism

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



G1



G2

I randomly choose G1 or G2, randomly scramble the vertices, and send the resulting adjacency matrix!

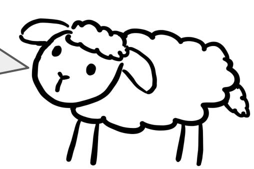
I say whether it's a scrambling of G1 or G2

	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



This is reminiscent of the pen example

A ZK Proof for Graph Non-Isomorphism

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:

- **"Soundness"**: If G1 and G2 are isomorphic, then Merlin "fools" Arthur into accepting with probability ≤ ½:

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

A ZK Proof for Graph Non-Isomorphism

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.***

- **"Efficiency"**: Arthur runs in polynomial time

Randomized Verifier is Essential

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

- We can assume 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.
- Let C be the message from Merlin.
- Given C, Arthur can decide whether the input is a YES or NO instance of L.
- 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**.

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 their own. So, it is not useful at all.

Admin

Exam Review Session led by Daphne:
Wednesday April 24th, 7-9pm, BBB 1670.

HW11 is due Tuesday 4/23 at 8pm.

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

Next Monday: Last Lecture

Nicole Wein: **Online Algorithms**

Thatchaphol Saranurak: **Vertex Connectivity**

Chris Peikert: **Quantum-Secure Cryptography with Lattices**

Mark Brehob: **Caching**

See piazza post for more detail