COMS BC3262: Introduction to Cryptography

Lecture 21: More on Signatures and Zero-Knowledge

April 15, 2026

Logistics

- **Extra credit opportunity**: Olive's talk tomorrow at 12pm in Milstein 402
  - No explicit point value, but if you're near a cutoff I'll use this to determine if you get bumped up
- HW 4 due tomorrow
- HW 5 released on Wednesday

Upcoming office hours

- My office hours are today 3pm-5pm
- I will not have office hours next week while I am traveling
- Mark and Tony should be having their office hours as usual
- See EdStem for any updates

Last Time

- Identification scheme
- Fiat-Shamir Transform
- Signature Schemes
  - DL-based signatures (Schnorr, DSA/ECDSA)

Today

- Interactive Proof Systems
  - Zero-knowledge proofs
- Maybe Lamport Signatures?

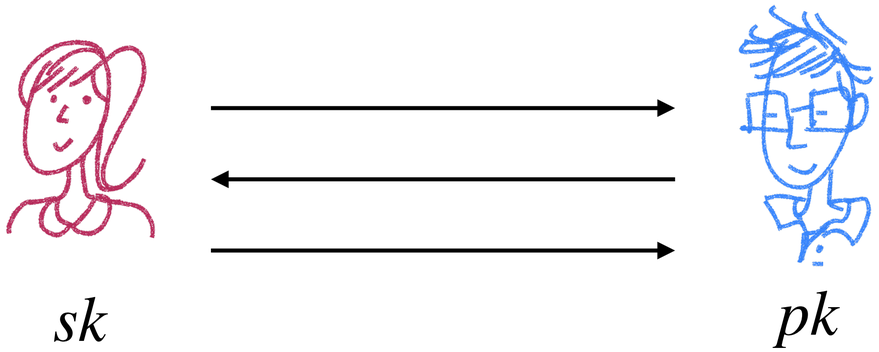
Interactive Proof Systems

What is a Proof?

- Informally, a proof is whatever convinces you of a conclusion
  - A proof has some starting point and contains sequence of agreed upon steps to reach a convincing conclusion
- However, what if we make a mistake or lie in our proof?
  - Proofs are not meaningful if no one checks the proof!
  - The verifier's job is to *not* be convinced if there was a mistake or lie
- The person trying to convince you is the prover
- The verifier is the one doubting the prover and checking their claims

Last Time: Identification Scheme

- Alice wants to prove to Bob she knows the sk associated with pk
- If Eve observes the communication (possibly many times), she can't later impersonate Alice
  - Eve should not be able to convince Bob she knows sk
  - Bob can't later impersonate Alice either



Properties of Proofs

- There are two general properties I want from any kind of proof (informally):
  - If the statement is true, I can convince you ("Completeness")
  - If the statement is false, I cannot convince you ("Soundness")

- What if I get lucky and happen to convince you of a false statement?

Handling Soundness Errors

- In experimental sciences, someone may run many experiments and gather a ton of evidence
  - Suppose each individual experiment has some large chance of a false positive
  - If you run enough of them, what are the chances *all of them* are false positives?
    - With enough experiments, it's astronomically small!
    - We can also do this with math!

Properties of Proofs

- There are two general properties I want from any kind of proof (informally):
  - If the statement is true, I can convince you ("*Completeness*")
  - If the statement is false, I cannot convince you ("*Soundness*")
- In the context of an experiment:
  - If the statement is **true**, the experiment always succeeds
  - If the statement is **false**, the experiment sometimes succeeds, but often it fails

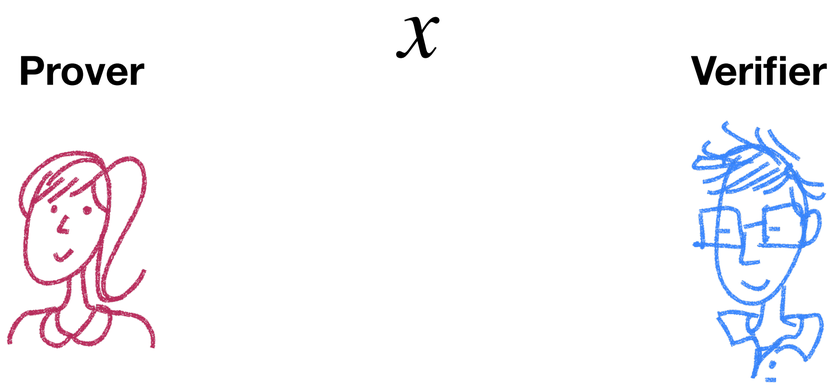
Properties of Proofs

- In the context of an experiment:
  - If the statement is **true**, the experiment always succeeds
  - If the statement is **false**, the experiment sometimes succeeds, but often it fails
- Putting these together:
  - Doing an experiment *once* might not tell us if something is true
  - But if we repeat it sufficiently many times:
    - If it's **true**, it'll always work
    - If it's **false**, it'll almost certainly fail at least once

Interactive Proof Systems

Scenario:

A prover P wants to convince a verifier V that a statement x is in a language L (i.e. $x \in L$)



Interactive Proof Systems

Scenario:

A prover P wants to convince a verifier V that a statement x is in a language L (i.e. $x \in L$)

x

Prover

Verifier

How does P convince V?

Maybe they send messages back and forth

(running an experiment, to use the previous comparison)

and at the end V decides if he's convinced.

Interactive Proof Systems

Scenario:

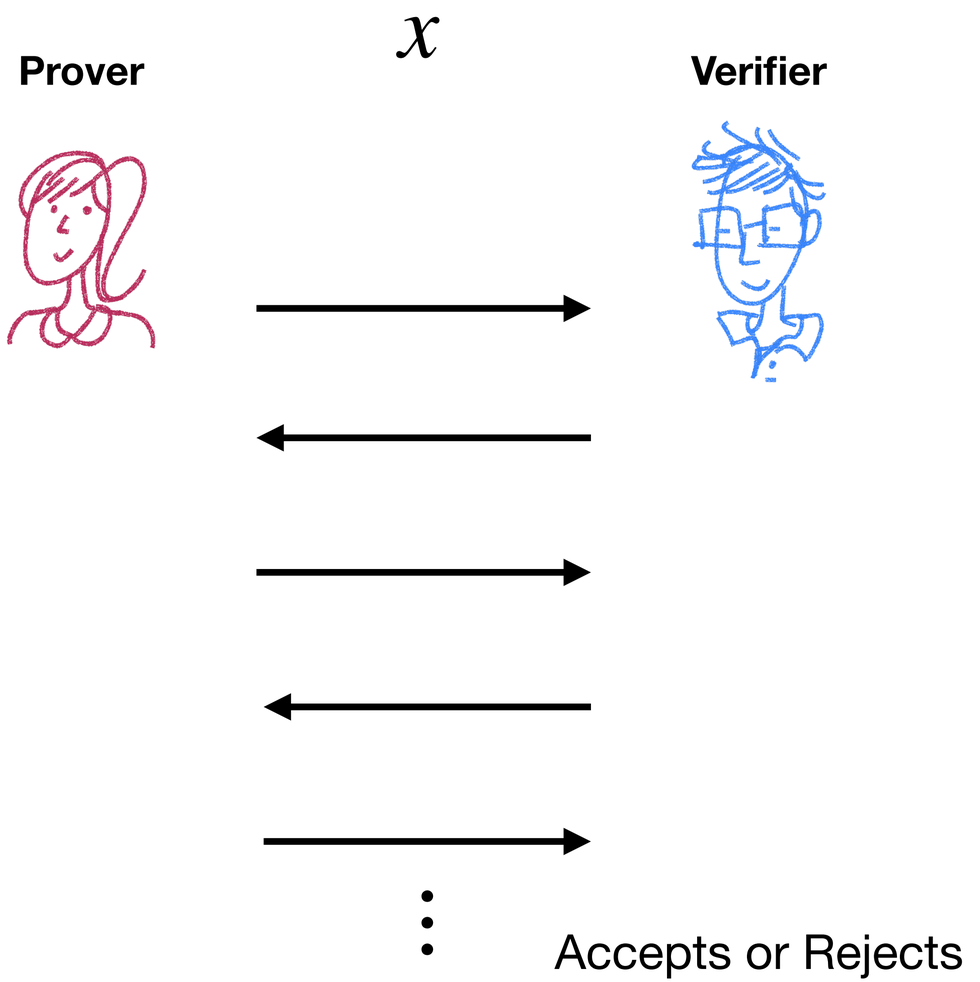
A prover P wants to convince a verifier V that a statement x is in a language L (i.e. $x \in L$)

How does P convince V?

Maybe they send messages back and forth

(running an experiment, to use the previous comparison)

and at the end V decides if he's convinced.



Interactive Proof Systems

Scenario:

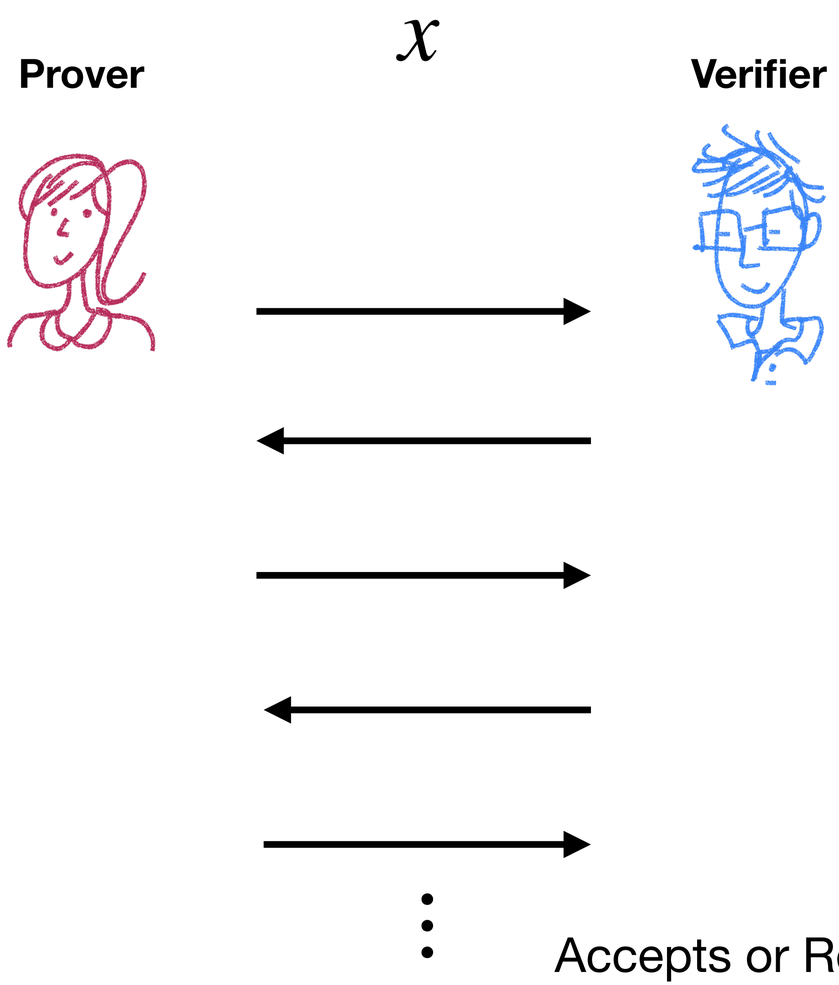
A prover P wants to convince a verifier V that a statement x is in a language L (i.e. $x \in L$)

How does P convince V?

Maybe they send messages back and forth
(running an experiment, to use the previous comparison)
and at the end V decides if he's convinced.

What if $x \notin L$?

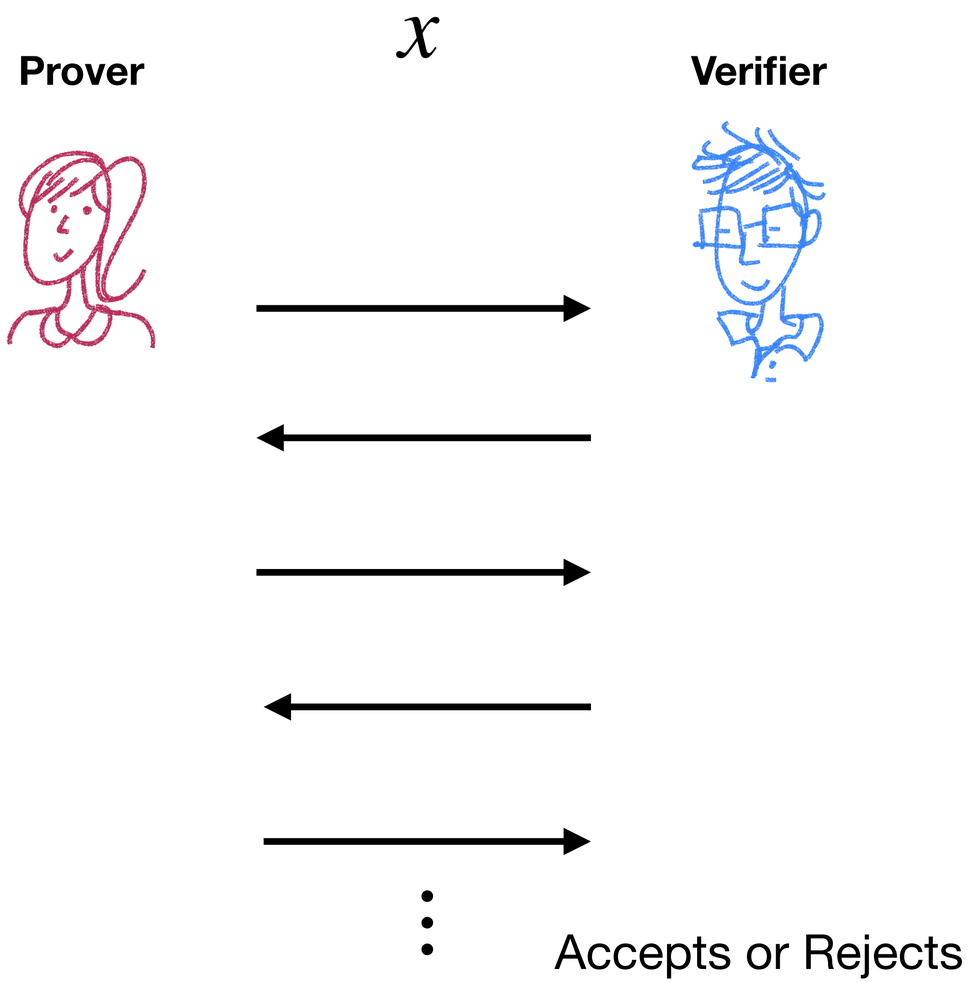
If $x \notin L$, then P^* will likely get caught lying



Interactive Proof Systems

Definition: A language $L \subseteq \{0,1\}^*$ has an interactive proof system if there exists a pair of interactive algorithms (P, V) where V is PPT and the following two properties holds:

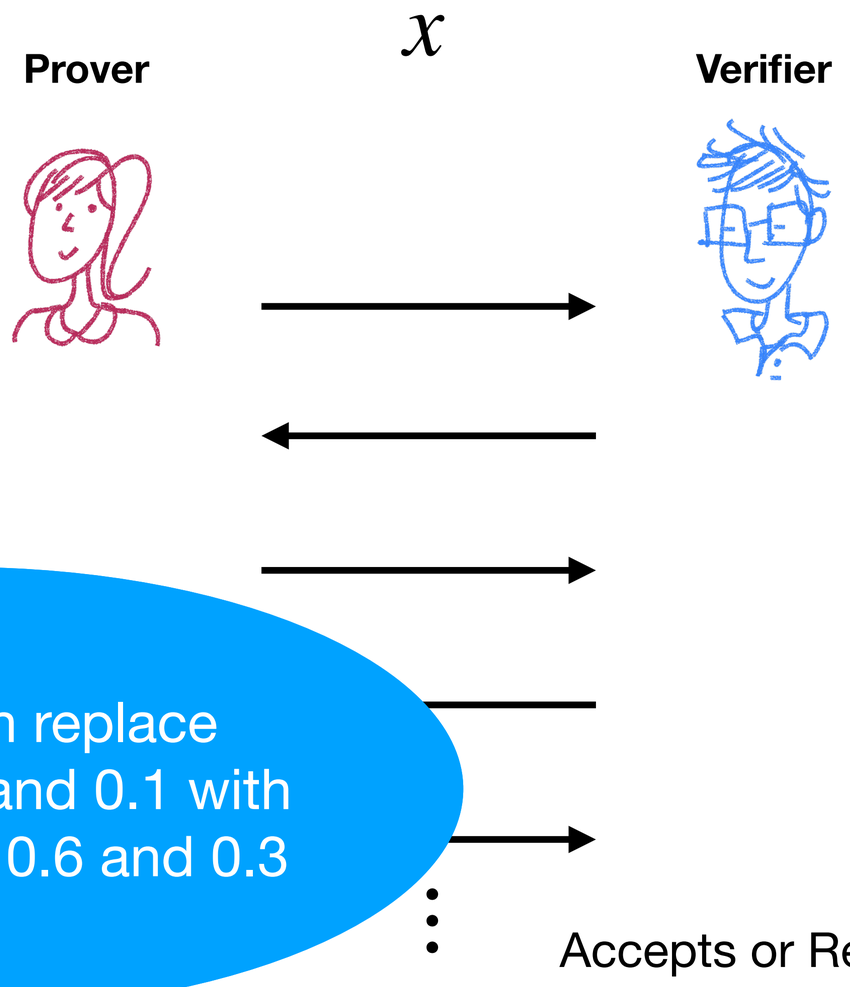
- Completeness: If $x \in L$, then the output of V after interacting with P on input x is 1 with probability ≥ 0.9
- Soundness: If $x \notin L$, then the output of V after interacting with a malicious P^* on input x is 1 with probability ≤ 0.1



Interactive Proof Systems

Definition: A language $L \subseteq \{0,1\}^*$ has an interactive proof system if there exists a pair of interactive algorithms (P, V) where V is PPT and the following two properties holds:

- Completeness: If $x \in L$, then the output of V after interacting with P on input x is 1 with probability ≥ 0.9
- Soundness: If $x \notin L$, then th
  after interacting with a malici
  input x is 1 with probability ≤ 0.1



Note we can replace constants 0.9 and 0.1 with anything up to 0.6 and 0.3

Interactive Proof Systems

Definition: A language $L \subseteq \{0,1\}^*$ has an interactive proof system if there exists a pair of interactive algorithms (P, V) where V is PPT and the following two properties holds:

- Completeness: If $x \in L$, then the output of V after interacting with P on input x is 1 with probability ≥ 0.9
- Soundness: If $x \notin L$, then the output of V after interacting with a malicious P^* on input x is 1 with probability ≤ 0.1

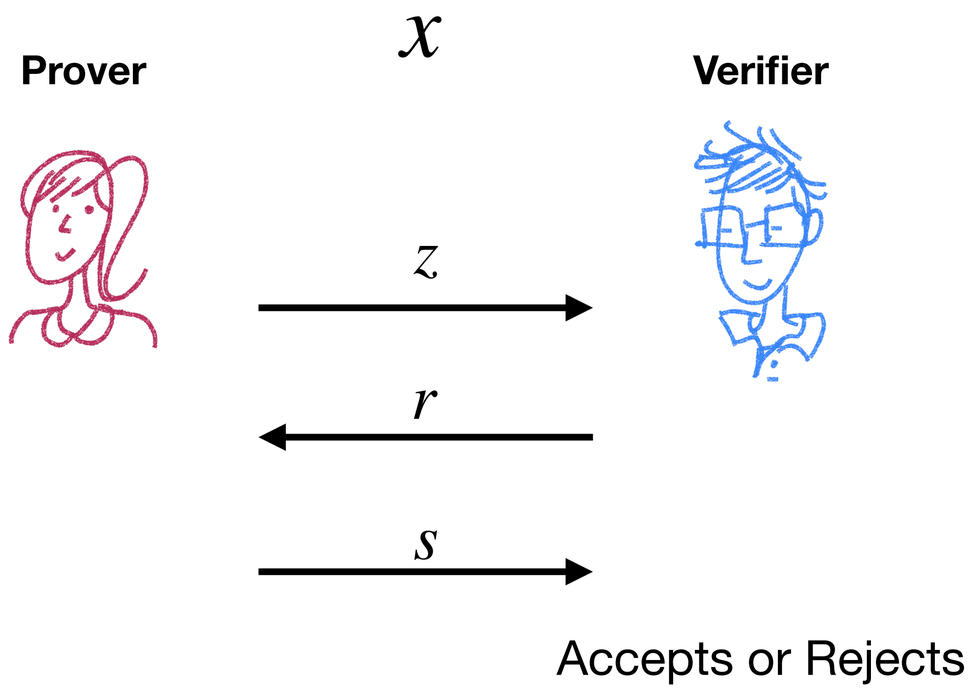
Notice we only require V be polynomially bounded.

If we require P^* also be PPT, this would be an **argument system**

Accepts or Rejects

Interactive Proof Systems

The three round structure we saw last time is known as a **Sigma protocol**



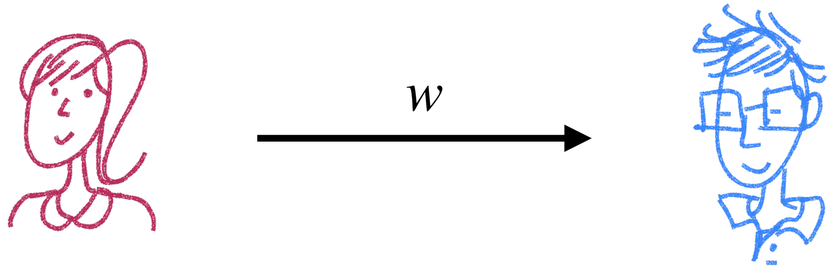
Interactive Proof Systems

IP is the set of languages that have interactive proof systems

Claim: $NP \subseteq IP$

Proof sketch:

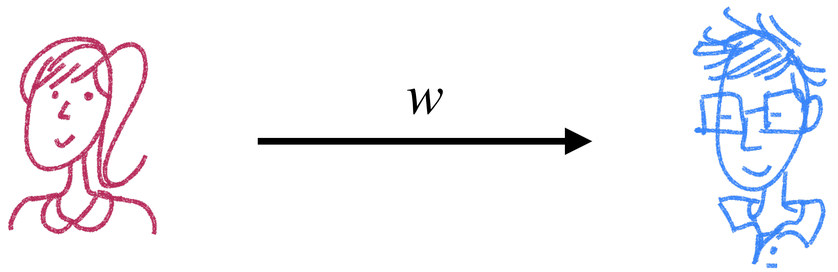
- If $L \in NP$ then $L = \{x \mid \exists w \text{ s.t. } R(x, w) = 1\}$ where checking R is efficient
- We can build an interactive protocol by having P send w to V and V checks $R(x, w)$



Zero Knowledge Proofs

Knowledge

- Knowledge enables you to do something
- You gain knowledge if you can do something new
  - If you go home after class and can't do anything new, you didn't gain any knowledge
- If I'm proving a statement and send you the witness, you are now able to do something new (prove the statement to other people)
  - Can I prove a statement without conveying any knowledge?

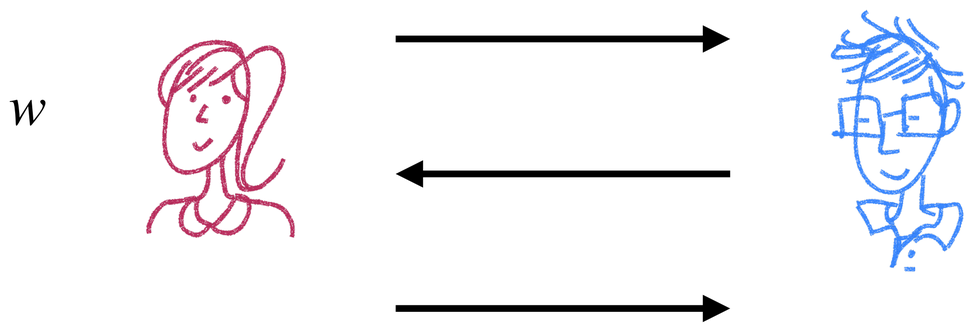


Zero-Knowledge

- The idea behind zero-knowledge is to convey/do something without the other person gaining the ability to do something new

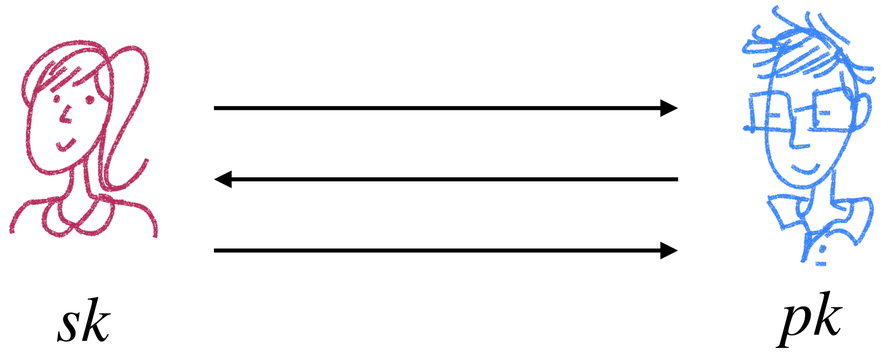
- With a zero-knowledge proof, a prover can convince a verifier that a statement is true without conveying anything beyond the fact it's true

  - In particular, the verifier is convinced but can't go to someone else and prove the statement themselves



Last Time: Identification Scheme

- Alice wants to prove to Bob she knows the sk associated with pk
- If Eve observes the communication (possibly many times), she can't later impersonate Alice
  - Eve should not be able to convince Bob she knows sk
  - Bob can't later impersonate Alice either



Identification Schemes are a zero-knowledge proof of knowledge of a secret key!

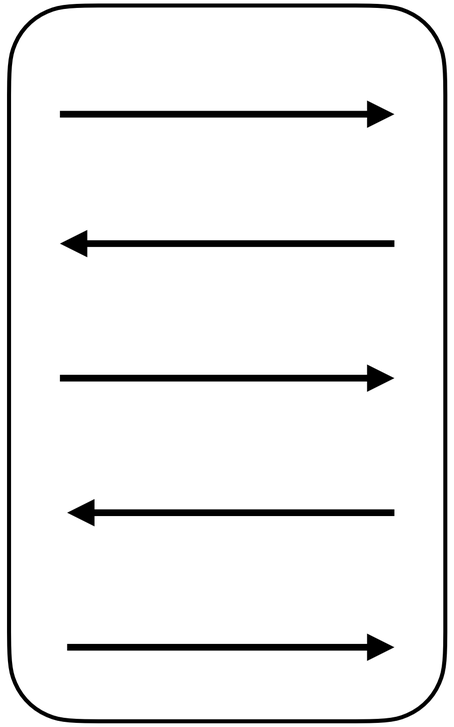
Zero-Knowledge Proofs

- How do we formalize zero-knowledge?
  - Earlier I described it as “the ability to do something new”
  - How can I ensure you aren’t able to do something new?
- Simulation paradigm!
  - You are able to simulate the interaction **on your own** what you would have seen if you had interacted with me (but without ever talking to me)
  - A simulator is an algorithm that produces a distribution of transcripts (messages sent back and forth) that is indistinguishable from transcripts produced by the real protocol

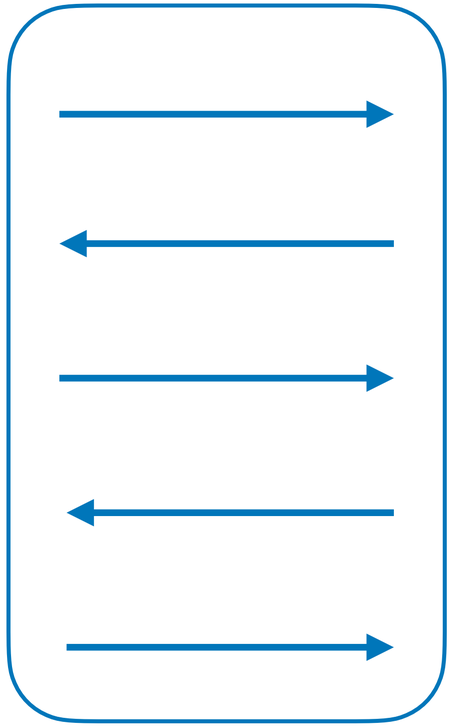
Simulation

x

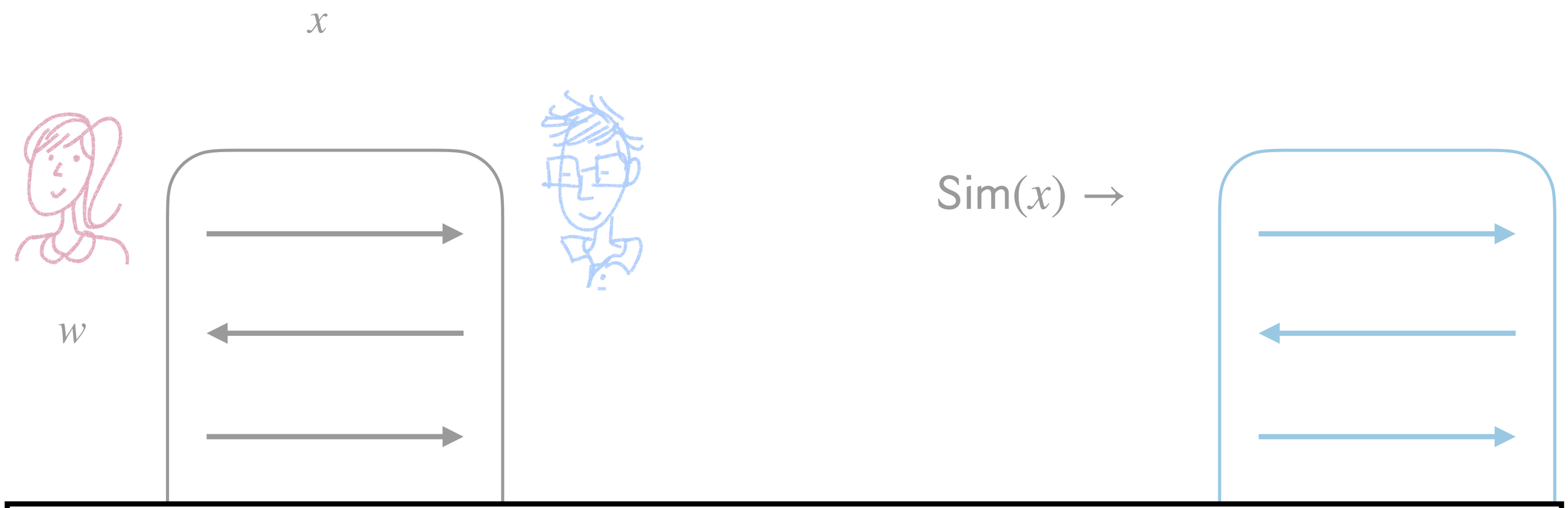
w



$\text{Sim}(x) \rightarrow$



Simulation

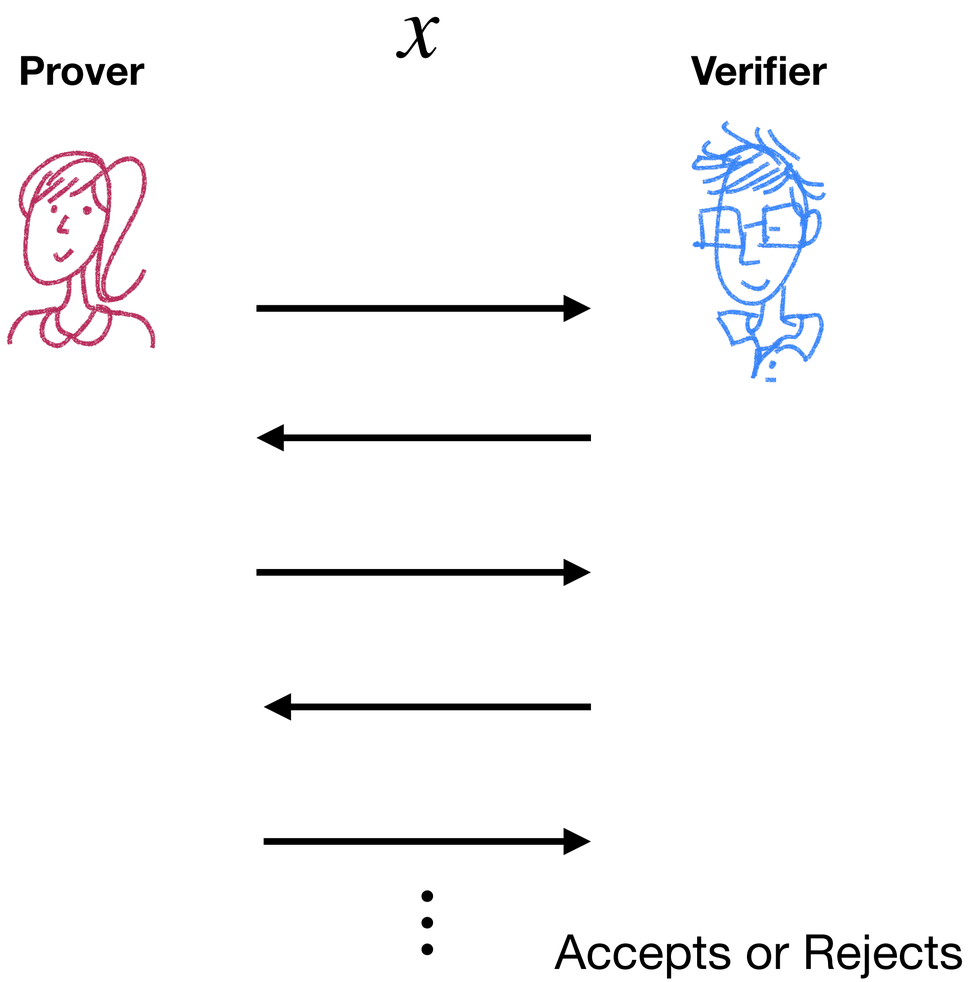


If you're able to produce a transcript on your own that looks indistinguishable from the one produced when talking to Alice, then you didn't gain anything new from the interaction with Alice!

Zero-Knowledge Interactive Proof Systems

Definition: Language L has a zero-knowledge interactive proof system if L has an interactive proof system and additionally satisfies:

- Zero-knowledge: For all PPT V^*, there exists a simulator S^* such that if $x \in L$, then the view of V^* when interacting with P on $x \approx S^*(x)$, where $\approx$ can be:
  - Equal (perfect zk)
  - Statistically close (statistical zk)
  - Computationally indistinguishable (computational zk)

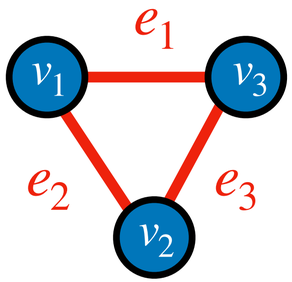


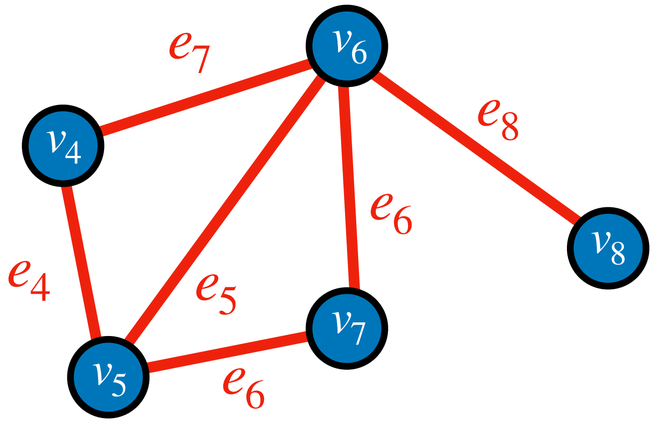
Graph 3-Coloring

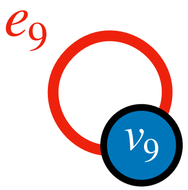
The beautiful animated slides are courtesy of my academic brother Jack Doerner

What is a Graph?

- A Graph is a collection of *Vertices*, connected by *Edges*. Every edge connects exactly two vertices, and vertices may have many edges.
- Usually we will give the vertices and edges unique labels $v_1, v_2, \ldots$ and $e_1, e_2, \ldots$
- The set of all vertices will be V and the set of all edges is E, and the graph is $G = (V, E)$.
- An edge is just a pair of vertices! So e.g. $e_2 = (v_1, v_2)$, $e_6 = (v_5, v_7)$, and $e_9 = (v_9, v_9)$.
- A Graph is a *logical structure*. The position of the vertices on the page doesn't matter, nor do the values of the labels. All that matters is how the vertices are connected (or not connected).



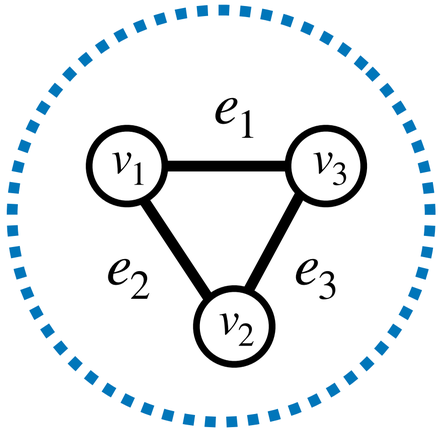


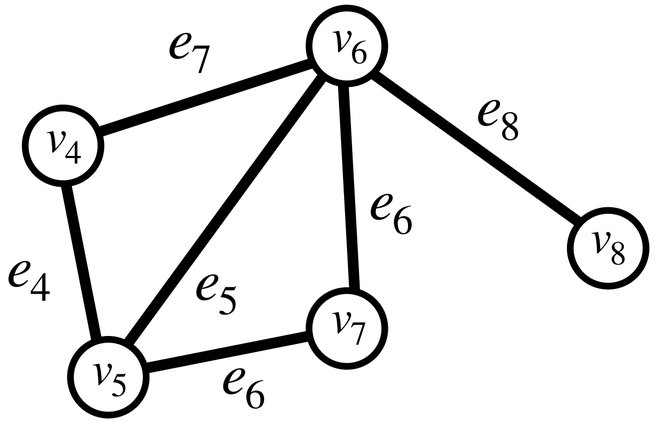


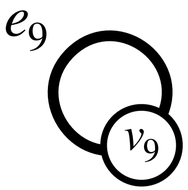
Coloring Graphs

- We say that a graph is k-*colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

Can you 2-color this one? **No.**
What about 3-coloring it?



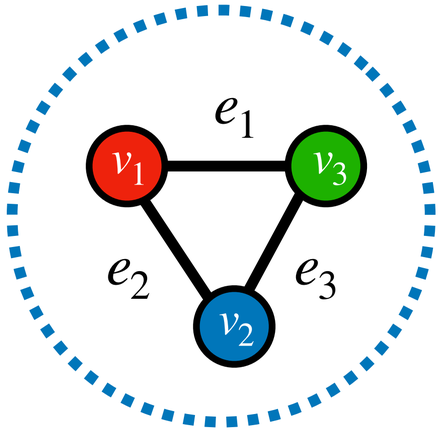


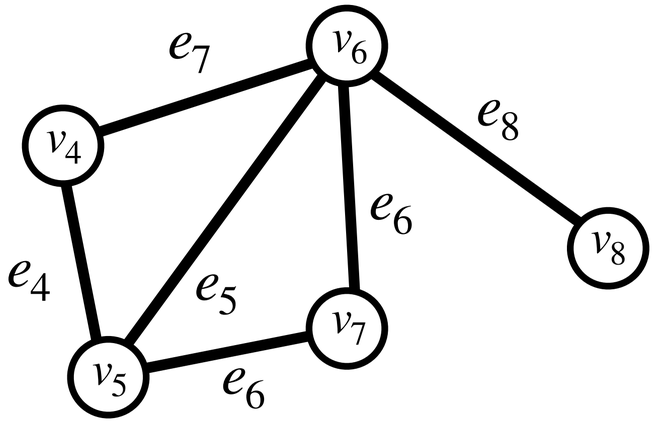


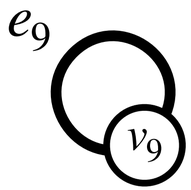
Coloring Graphs

- We say that a graph is k-*colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

Can you 2-color this one? **No.**
What about 3-coloring it? **Yes.**



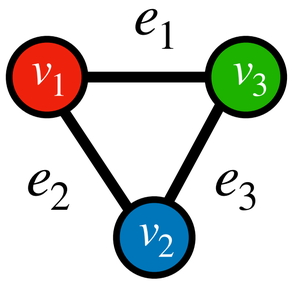


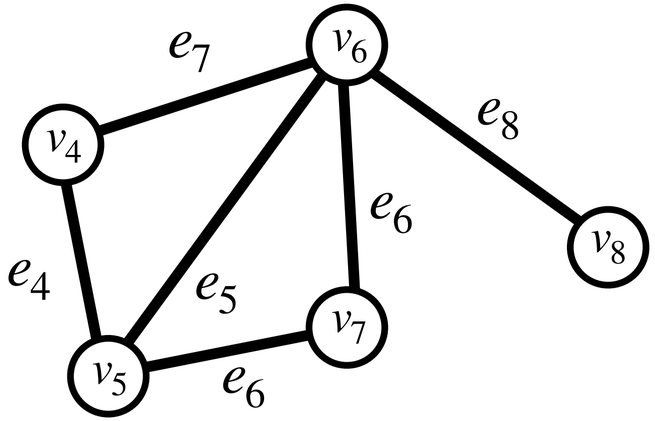


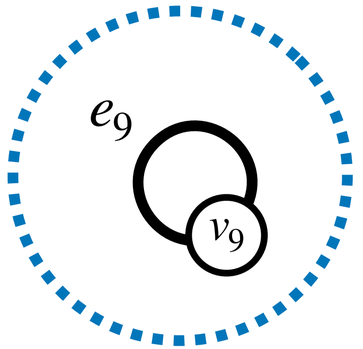
Coloring Graphs

- We say that a graph is k-*colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

What about this one?
It can't even be 1-colored!



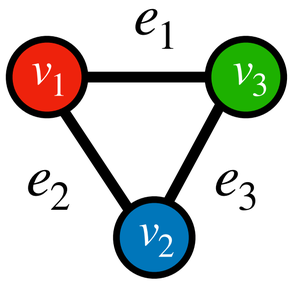


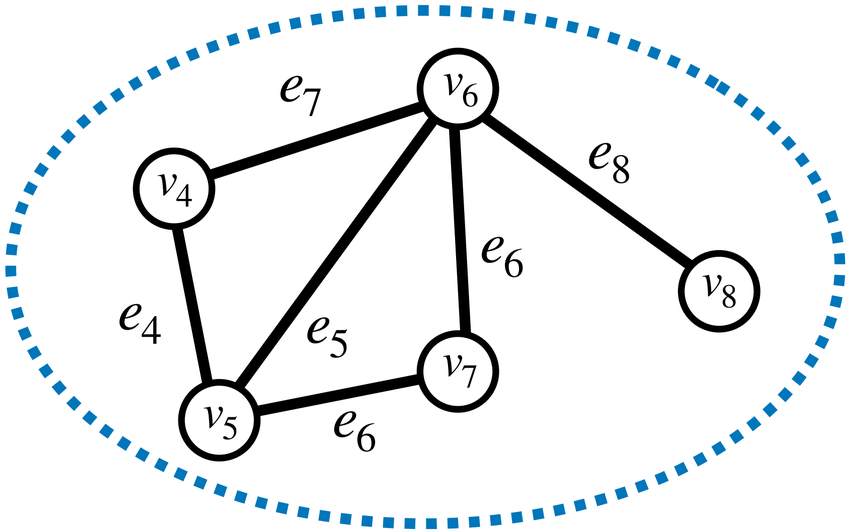


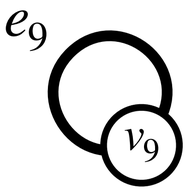
Coloring Graphs

- We say that a graph is k-*colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

What's the smallest number of colors for this one?



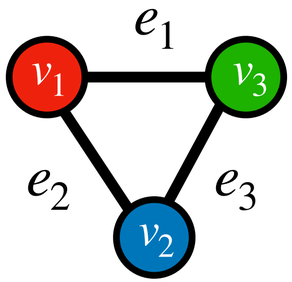


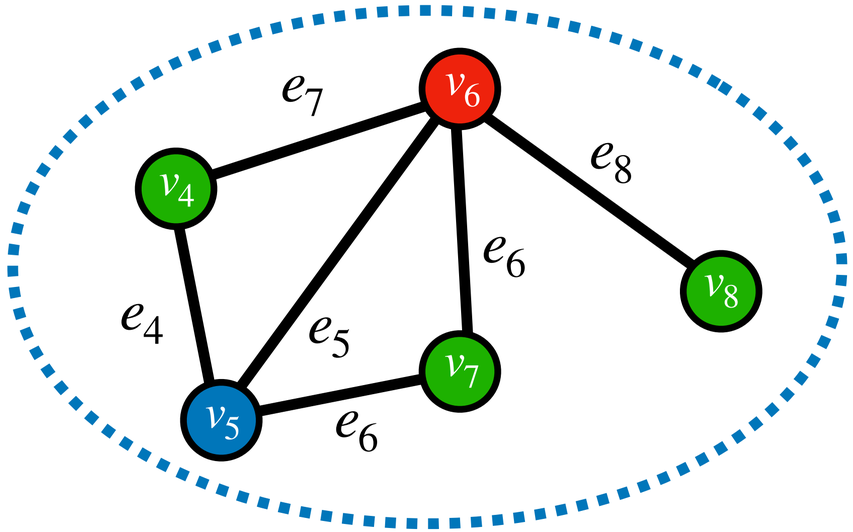


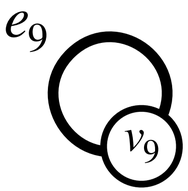
Coloring Graphs

- We say that a graph is k-*colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

What's the smallest number of colors for this one? **3**



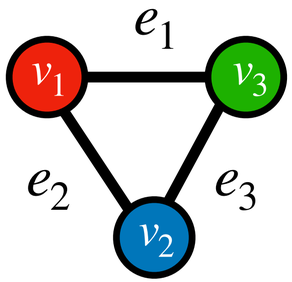


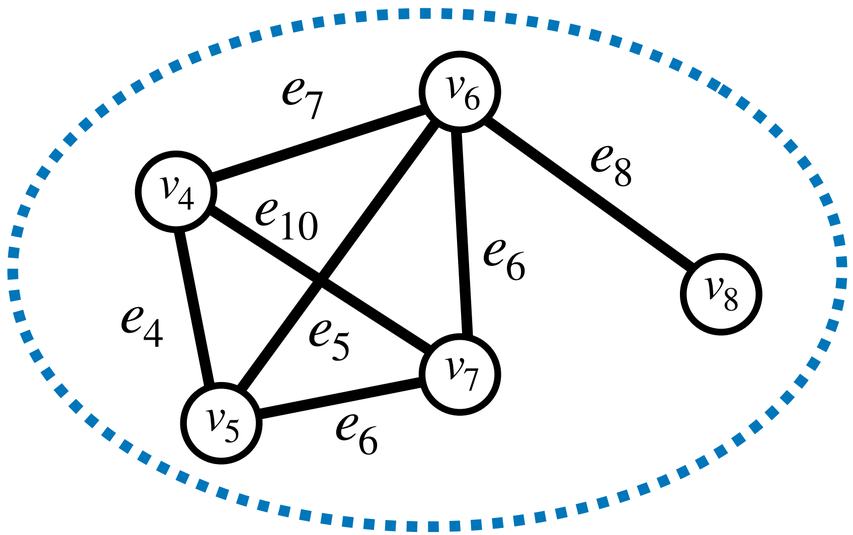


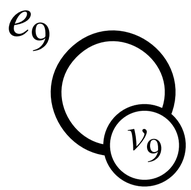
Coloring Graphs

- We say that a graph is k-*colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

But if I add one more edge?



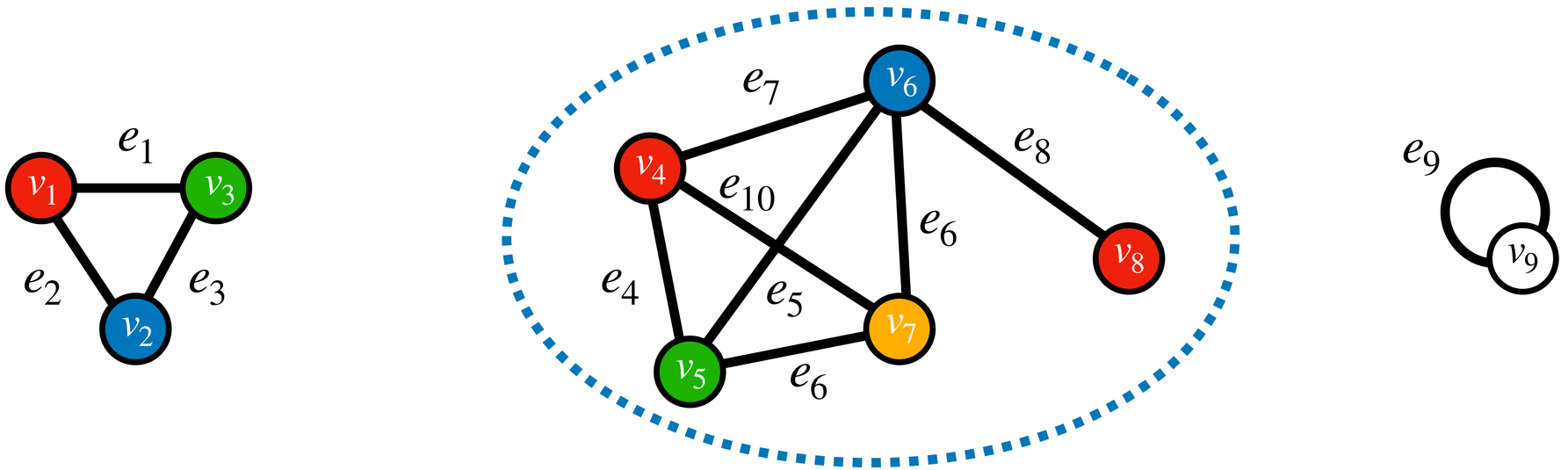




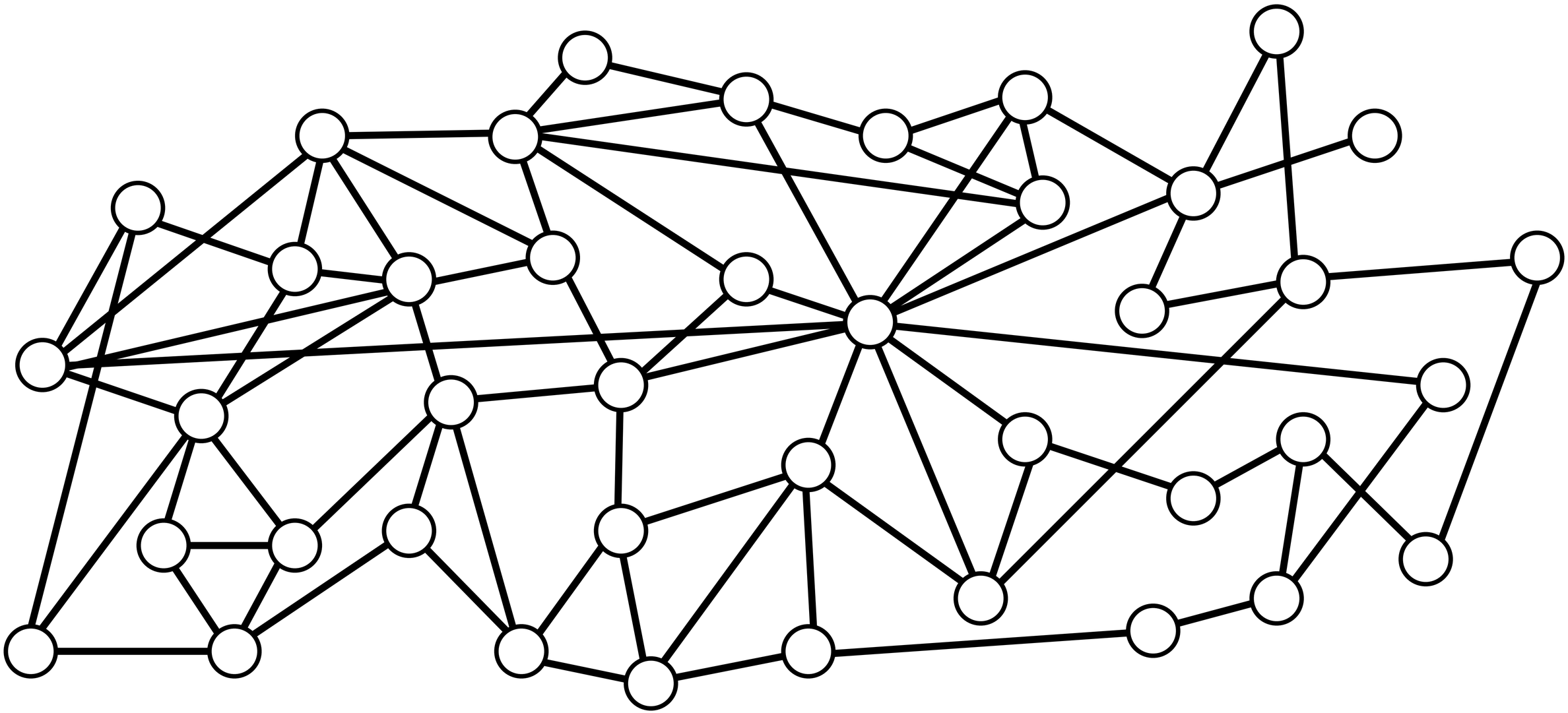
Coloring Graphs

- We say that a graph is k-*colorable* if there exists a set of k colors such that:
  1. Every vertex is assigned one of the colors. This assignment is called a *coloring*.
  2. Every edge connects two vertices of *different* colors. If this is true the coloring is *proper*.
- Not every graph can be k-colored for every number k.

But if I add one more edge? **4**



Can you 3-color this graph? It’s hard, right?



Can you 3-color this graph?

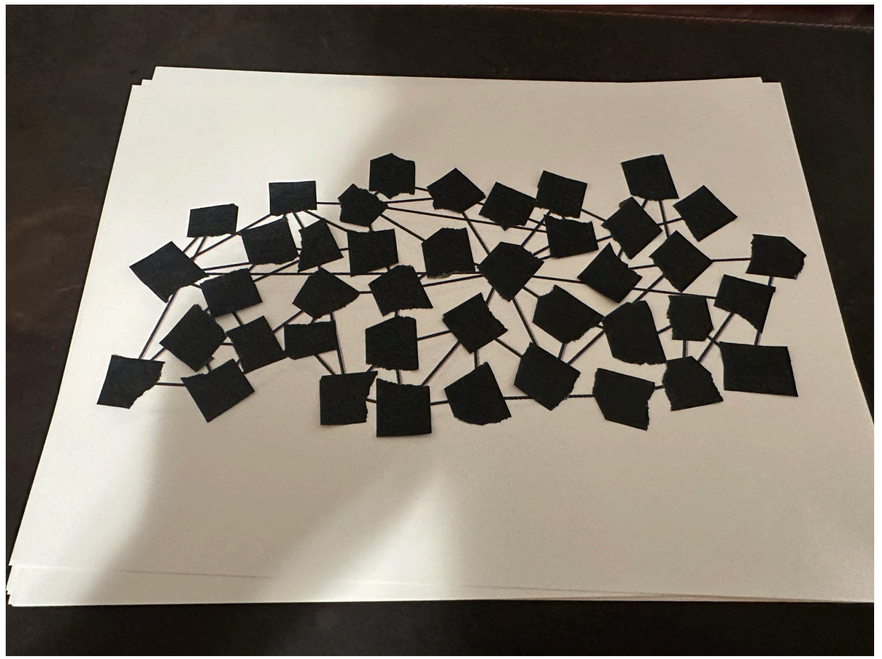
- This question is not just hard… It's NP-complete!
- I want to prove to you that for this graph, the answer is "Yes, it can be 3-colored".
- However, if I just show you the coloring, then I won't be able to ask you to find it on an exam later!
- Instead, I will prove to you that this graph can be 3-colored without increasing your knowledge.

Zero-Knowledge Interactive Proof for 3-Coloring

- In order to convince you that the graph is 3-colorable, we'll participate in an interactive protocol
    - I am the prover and you are the verifier
- You have to be careful because I might try to cheat in the protocol and deceive you into believing a false statement
    - If I'm lying, by soundness you have a good chance of catching me
- But if I'm acting honestly, by completeness you'll never think I'm cheating

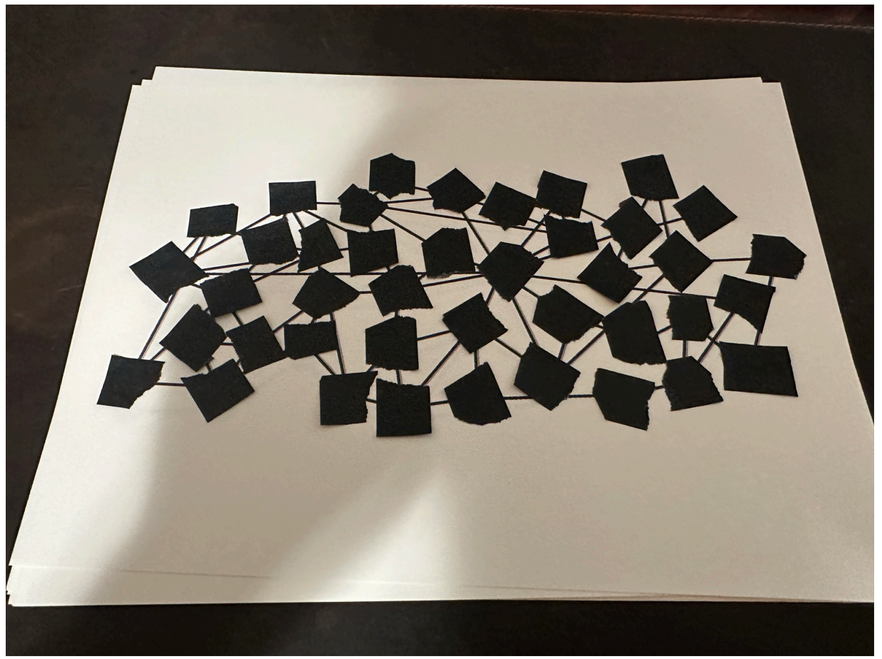
Zero-Knowledge Interactive Proof for 3-Coloring

- First, I'll randomize the coloring by permuting the colors (so e.g. red becomes green, green becomes blue, blue becomes red)
- Then, I'll **commit** to my coloring by sending you something corresponding to each vertex that is both hiding and binding.
  - Hiding: you don't know what is being committed to until I reveal it later
  - Binding: When I reveal the value to you later, I can't change it



Zero-Knowledge Interactive Proof for 3-Coloring

- First, I'll randomize the coloring by permuting the colors (so e.g. red becomes green, green becomes blue, blue becomes red)
- Then, I'll **commit** to my coloring by sending you something corresponding to each vertex that is both hiding and binding.
- You can pick any edge to reveal the colors of the vertices it connects
  - If the two colors are different, we're happy
  - If the two colors are the same (or some 4th color), you caught me lying!



Zero-Knowledge Interactive Proof for 3-Coloring

- **Completeness**: What's the likelihood you catch me lying when I'm telling the truth?
- **Soundness**: What's the likelihood you catch me lying when I'm lying?
  - If I don't know a 3-coloring, then there must either be at least one edge that has the same color on both ends or an invalid color on one end.
  - There are 77 edges in the graph. If you pick an edge randomly, what is the probability you catch me cheating?
    - At least $1/77$
    - The probability I get away with it each time is at most $76/77$

Zero-Knowledge Interactive Proof for 3-Coloring

- Let’s suppose we repeat this game r times

Zero-Knowledge Interactive Proof for 3-Coloring

- Let's suppose we repeat this game r times
- In order to fool you, I need to not get caught cheating all r times.
- What is the probability I don't get caught in all repetitions?
  - $\left(\frac{76}{77}\right)^r$
    - What if $r = 881$? That's less than $1/1000000$
    - What if $r = 1591$? Thats less than 2^{-30}

Zero-Knowledge Interactive Proof for 3-Coloring

- **Zero-knowledge**: How might you be able to simulate this?
  - Given only the public components, how can an algorithm produce a transcript that looks indistinguishable from a real run of the protocol?

How to construct a commitment?

- We want two properties from a commitment:
  - Hiding: The commitment does not reveal the value being committed
  - Binding: The commitment can only be opened to the original value that was committed to
- Are there any primitives we've seen so far that would enable us to do this?
  - One-way functions!!

Theorem [GGM]: If OWF exist, then every $L \in NP$ has a computational zk proof system

Lamport Signatures

Minimal Assumptions for Signatures

- Speaking of OWFs…
- We saw signatures from RSA and DL in the random oracle model
  - Can we use weaker assumptions?
- One-time signatures can be constructed from OWFs!
  - One-time signatures are defined like regular signatures but the adversary can only see a single signature before having to produce a forgery

Lamport's Signatures

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a OWF. We construct a signature for n-bit messages as follows.

- Gen(1^n): Sample $x_1^0, x_1^1, \ldots, x_n^0, x_n^1 \leftarrow \{0,1\}^n$ and compute $y_i^b = f(x_i^b)$. Output $pk = (y_1^0, y_1^1, \ldots, y_n^0, y_n^1)$ and $sk = (x_1^0, x_1^1, \ldots, x_n^0, x_n^1)$

- Sign$_{sk}(m)$: Let $m = m_1, \ldots, m_n$. Output the signature $\sigma = (x_1^{m_1}, \ldots, x_n^{m_n})$

- Verify$_{pk}(m, \sigma)$: Denote $\sigma = \sigma_1, \ldots, \sigma_n$. Output 1 if $y_i^{m_i} = f(\sigma_i)$ for every i. Otherwise output 0.

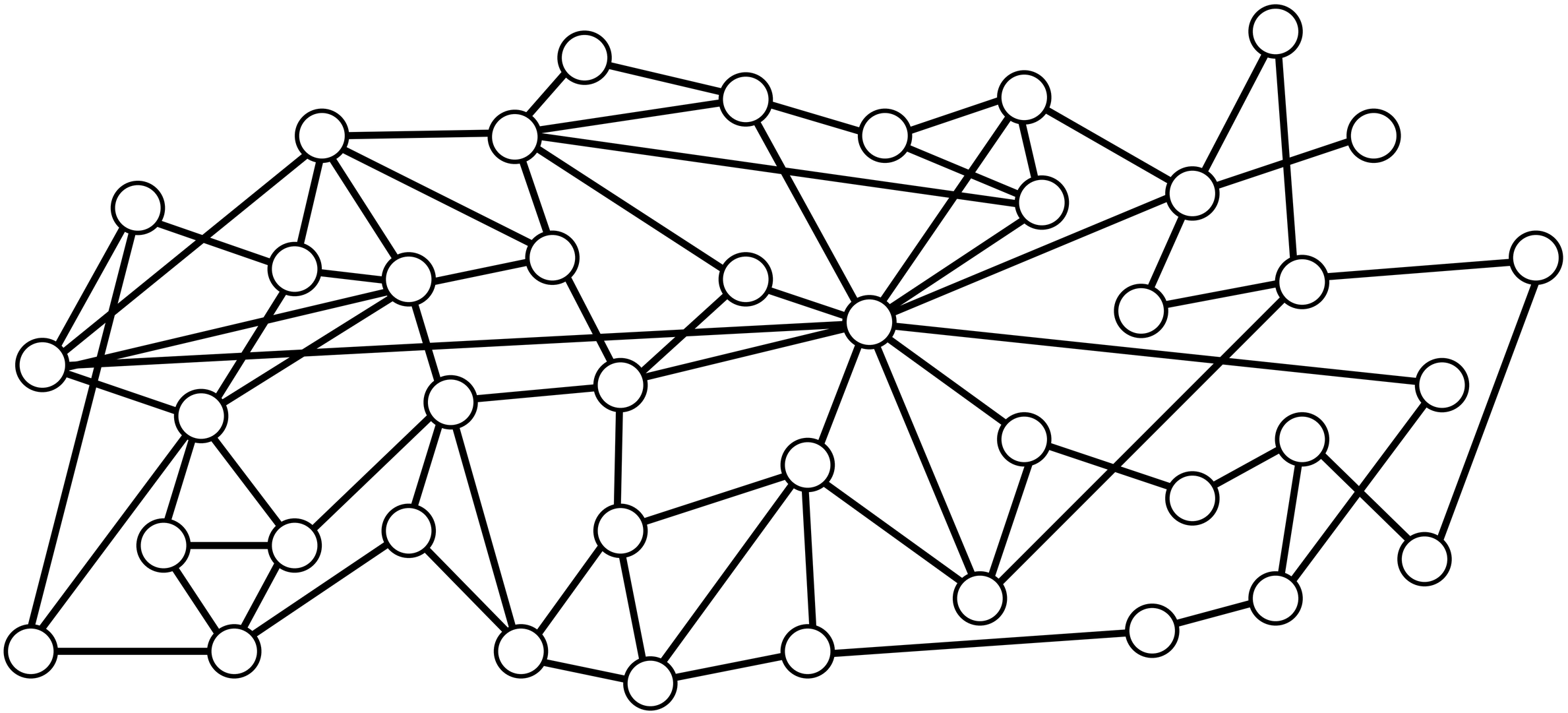
Lamport’s Signatures

- Can be extended to arbitrary length using hash functions (hash-and-sign)
- Can go beyond “one-time usage” by either:
  - Chain-based signatures (stateful)
  - Tree-based signatures (stateless)
- Entire construction is based only on one-way functions but proof is involved
- Regained attention due to post-quantum signatures

Next Time

- Guest lectures!

PS. Remember this Question?



It won’t be on the exam :)