Focus thus far in the course: protecting communication (e.g., message confidentiality and message integrity)

Remainder of course: protecting <u>computations</u>

<u>Zero-knowledge</u>: a defining idea at the heart of theoretical cryptography

↳ Idea will seem very counter-intuitive, but surprisingly powerful ← with surprising implications (DSA/ECDSA signatures based on ZK!)

↳ Showcases the importance and power of definitions (e.g., "What does it mean to know something?")

We begin by introducing the notion of a "proof system"

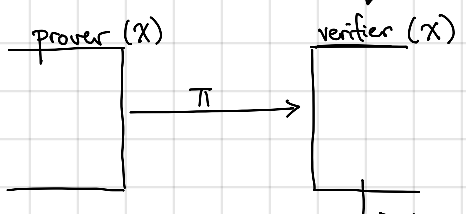
- <u>Goal</u>: A prover wants to convince a verifier that some <u>statement</u> is true

 e.g., "This Sudoku puzzle has a unique solution"
 "The number N is a product of two prime numbers p and q"
 "I know the discrete log of h base g"

 } these are all examples of statements

We model this as follows:

the verifier is assumed to be an efficient algorithm



x: statement that the prover is trying to prove (known to both prover and verifier)

↳ We will write $\mathcal{L}$ to denote the set of <u>true</u> statements (called a "language")

π: the proof of x

↳ $b \in \{0,1\}$ – given statement x and proof π, verifier decides whether to <u>accept</u> or <u>reject</u>

Properties we care about:

- <u>Completeness</u>: Honest prover should be able to convince honest verifier of true statements

$$\forall x \in \mathcal{L} : \Pr[\pi \leftarrow P(x) : V(x,\pi) = 1] = 1$$

- <u>Soundness</u>: Dishonest prover cannot convince honest verifier of false statement

$$\forall x \notin \mathcal{L} : \Pr[\pi \leftarrow P(x) : V(x,\pi) = 1] < \frac{2}{3}$$

<u>Important</u>: We are <u>not</u> restricting to <u>efficient</u> provers

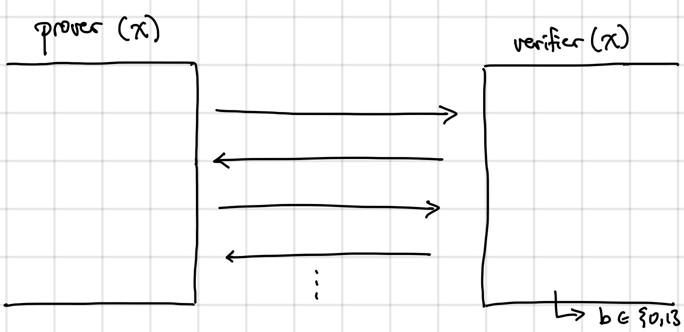
Typically, proofs are "one-shot" (i.e., single message from prover to verifier) and the verifier's decision algorithm is <u>deterministic</u>

↳ Languages with these types of proof systems precisely coincide with NP (proof of statement x is to send NP witness w)

Going beyond NP: we augment the model as follows

- Add randomness: the verifier can be a randomized algorithm
- Add interaction: verifier can ask "questions" to the prover

<u>Interactive proof systems</u> [Goldwasser-Micali-Rackoff]:



Set of languages that have an interactive proof system is denoted IP.

<u>Theorem (Shamir)</u>: IP = PSPACE ← languages that can be decided in polynomial <u>space</u> [very large class of languages!]

Takeaway: interaction and randomness is very useful

↳ In fact, enables a new property called zero-knowledge

Consider following example: Suppose prover wants to convince verifier that $N = pq$ where p, q are prime (and secret).

prover (N, p, q) verifier (N)

$\pi = (p, q)$ →

↓

accept if $N = pq$ and reject otherwise

Proof is certainly complete and sound, but now verifier also learned the factorization of N... (may not be desirable if prover was trying to convince verifier that N is a proper RSA modulus (for a cryptographic scheme) without revealing factorization in the process

↳ In some sense, this proof conveys information to the verifier [i.e., verifier learns something it did not know before seeing the proof]

Zero-knowledge: ensure that verifier does not learn anything (other than the fact that the statement is true)

How do we define "zero-knowledge"? We will introduce a notion of a "simulator."

Definition. An interactive proof system $\langle P, V \rangle$ for a language $\mathcal{L}$ is zero-knowledge if for all efficient (and possibly malicious) verifiers V^*, there exists an efficient simulator S such that for all $x \in \mathcal{L}$:

$$\underbrace{\text{View}_{V^*}(\langle P, V \rangle(x))}_{\text{random variable denoting the set of messages sent and received by } V^* \text{ when interacting with the prover } P \text{ on input } x} \overset{c}{\approx} S(x)$$

What does this definition mean?

$\text{View}_{V^*}(P \leftrightarrow V^*(x))$: this is what V^* sees in the interactive proof protocol with P

$S(x)$: this is a function that only depends on the statement x, which V^* already has

If these two distributions are indistinguishable, then anything that V^* could have learned by talking to P, it could have learned just by invoking the simulator itself, and the simulator output only depends on x, which V^* already knows

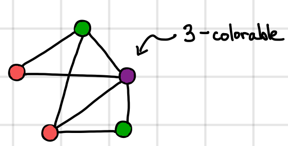
↳ In other words, anything V^* could have learned (i.e., computed) after interacting with P, it could have learned without ever talking to P!

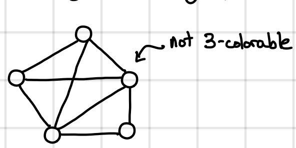
Very remarkable definition!

More remarkable: Using cryptographic commitments (← can in fact be constructed from OWFs), then every language $\mathcal{L} \in \mathrm{IP}$ has a zero-knowledge proof system.

↳ Namely, anything that can be proved can be proved in zero-knowledge!

We will show this theorem for NP languages. Here it suffices to construct a single zero-knowledge proof system for an NP-complete language. We will consider the language of graph 3-colorability.





3-coloring: given a graph G, can you color the vertices so that no adjacent nodes have the same color?