CSCI 361 Lecture 22:

Miscellaneous Lecture

Shikha Singh

Announcements & Logistics

- HW 7 and 8 graded feedback returned
  - Solutions on GLOW
- HW 9 grading in progress
- **HW 10** due tomorrow (Dec 3)
  - Questions on time and space complexity
  - **Optional** and grade will **replace** your lowest HW grade so far
  - Problems in it/topics are part of the syllabus for final exam

Announcements & Logistics

- Final exam in on **Dec 12 (Fri) at 1.30-3.30 pm in Schow 30A**
  - 2 hr closed-notes exam
  - Cumulative with an emphasis on decidability, undecidability, time complexity, NP hardness reductions, space complexity
  - Practice final will be released tomorrow
- Final Q&A/review session about practice exam or anything else
  - **Wed Dec 10**, **noon -1.30 pm** in Schow 30A
  - I'll order lunch/pizza!
  - Please go over assignments/practice exam over reading period and bring any questions you have

Plan for This Week

- Miscellaneous/Fun lecture today (not on the syllabus for the final)
- Short wrap up lecture on Thursday and SCS evals in class:
    - Please bring your laptop with you
    - Will leave time to fill out SCS evals in class

Today: Extra/ Fun Stuff

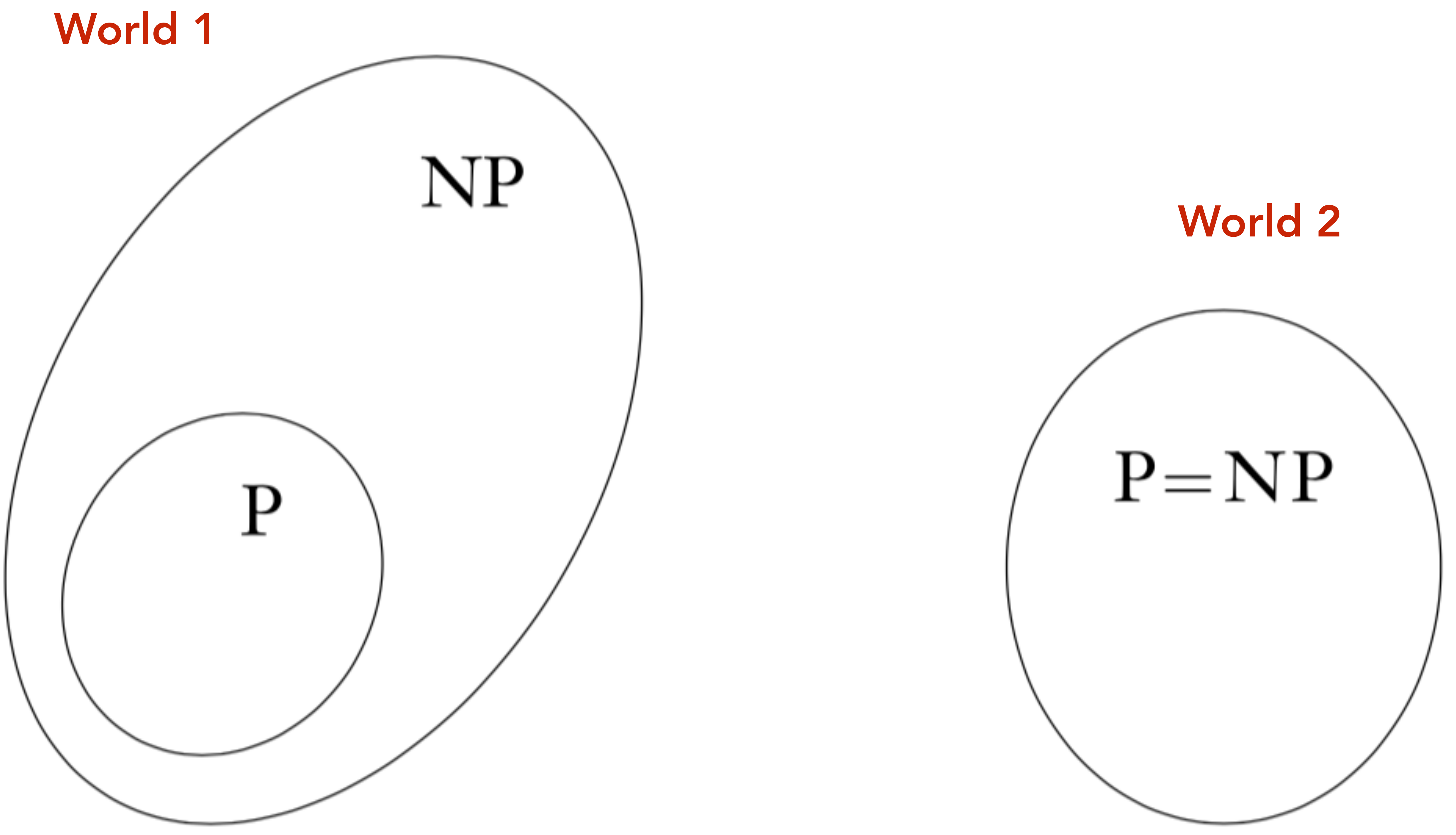
- Discuss the paper Impagliazzo's Five Worlds
  - P vs NP problem beyond the binary
- Two directions past beyond the models we discussed:
  - Does randomness help?
  - Does "interaction" help?
- If time permits, discuss the complexity theoretic capabilities of LLMs

[Part 1]

Impagliazzo's Five Worlds

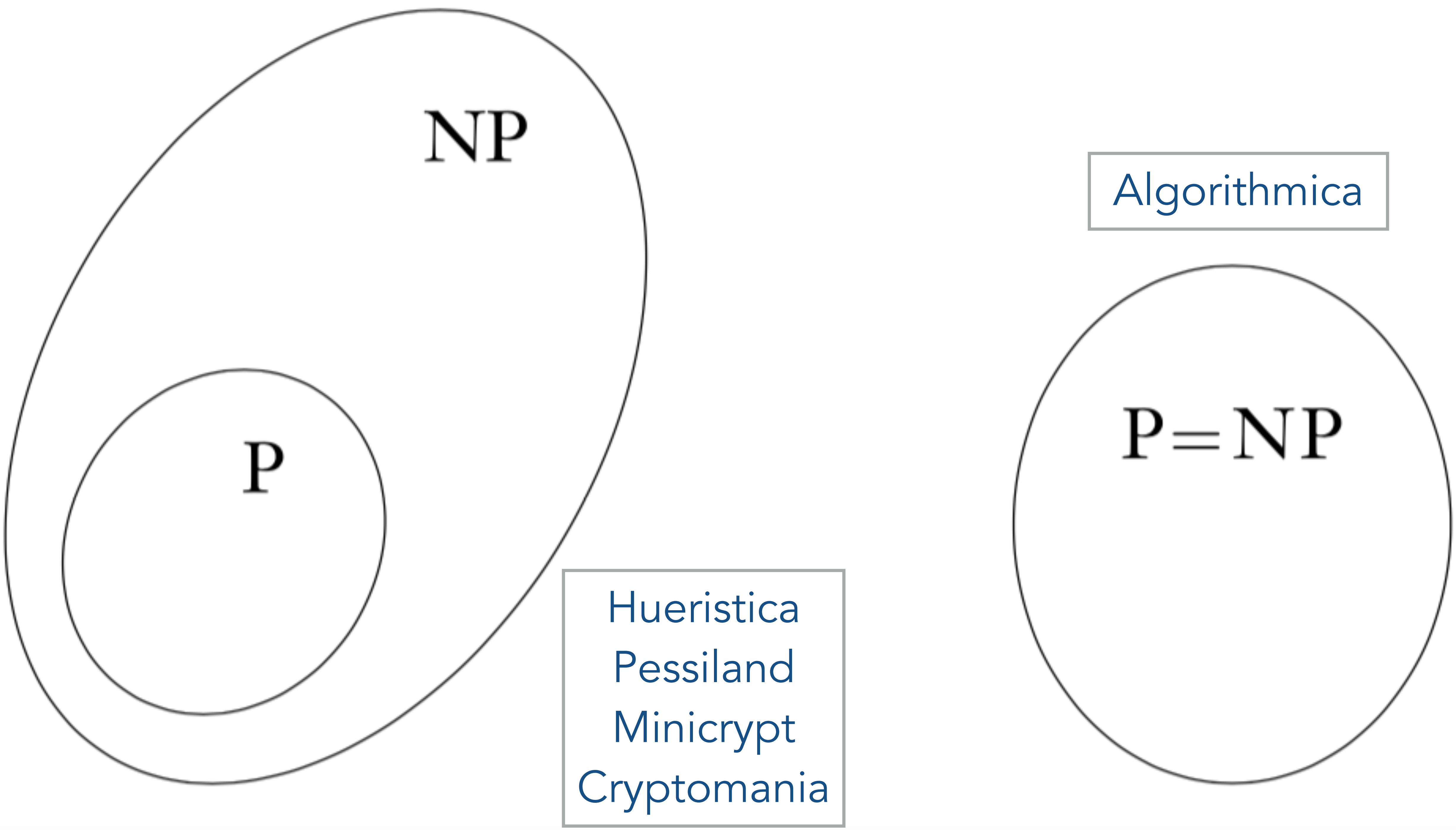
Impagliazzo, Russell. "A personal view of average-case complexity."
Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference. IEEE, 1995.

P vs NP: Binary Viewpoint



Impagliazzo's Five Worlds

Four Possible Worlds within P $\neq$ NP

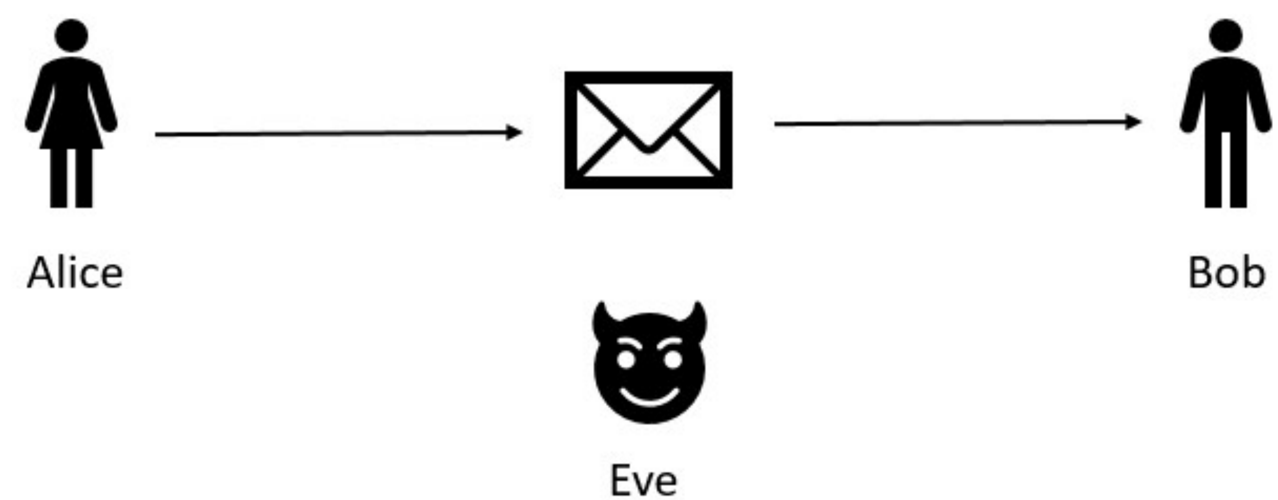


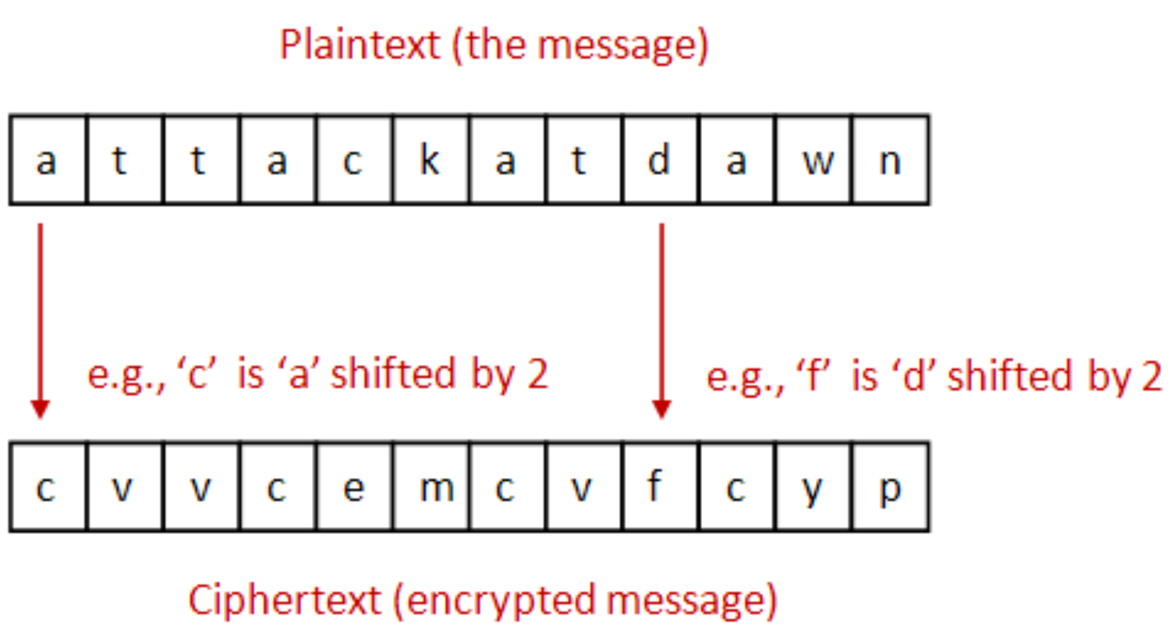
Impagliazzo's Five Worlds

- *"There is a large gap between a problem not being easy and the same problem being difficult."* - Russell Impagliazzo
- Different subworlds within $P \neq NP$ are based on *average-case* hardness rather than worst-case
  - Are hard instances easily found (sampled)?
- Some of these worlds require knowing about
  - One-way functions
  - Public-key cryptography

Cryptography Basics

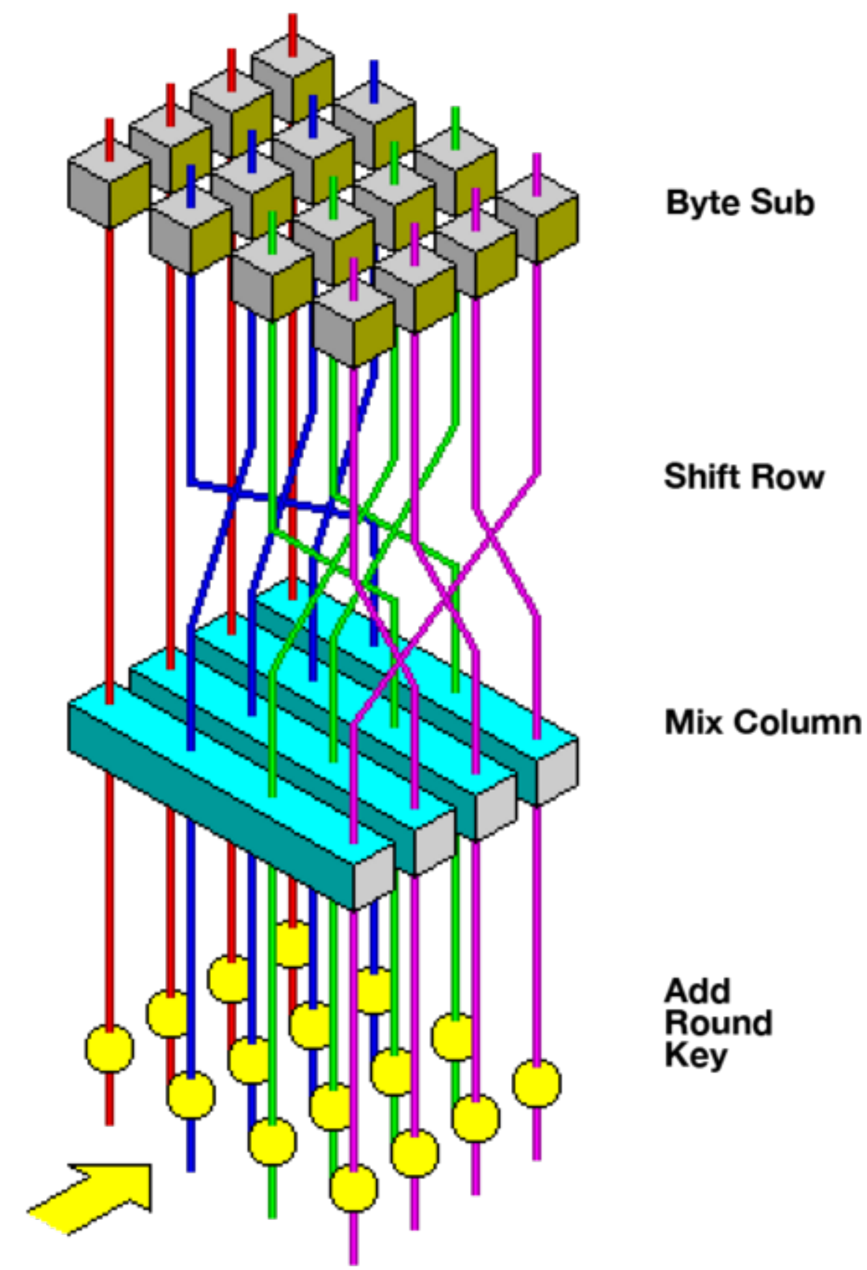
- Study of secure communication in the presence of adversaries
- Earliest use: simple Caesar ciphers





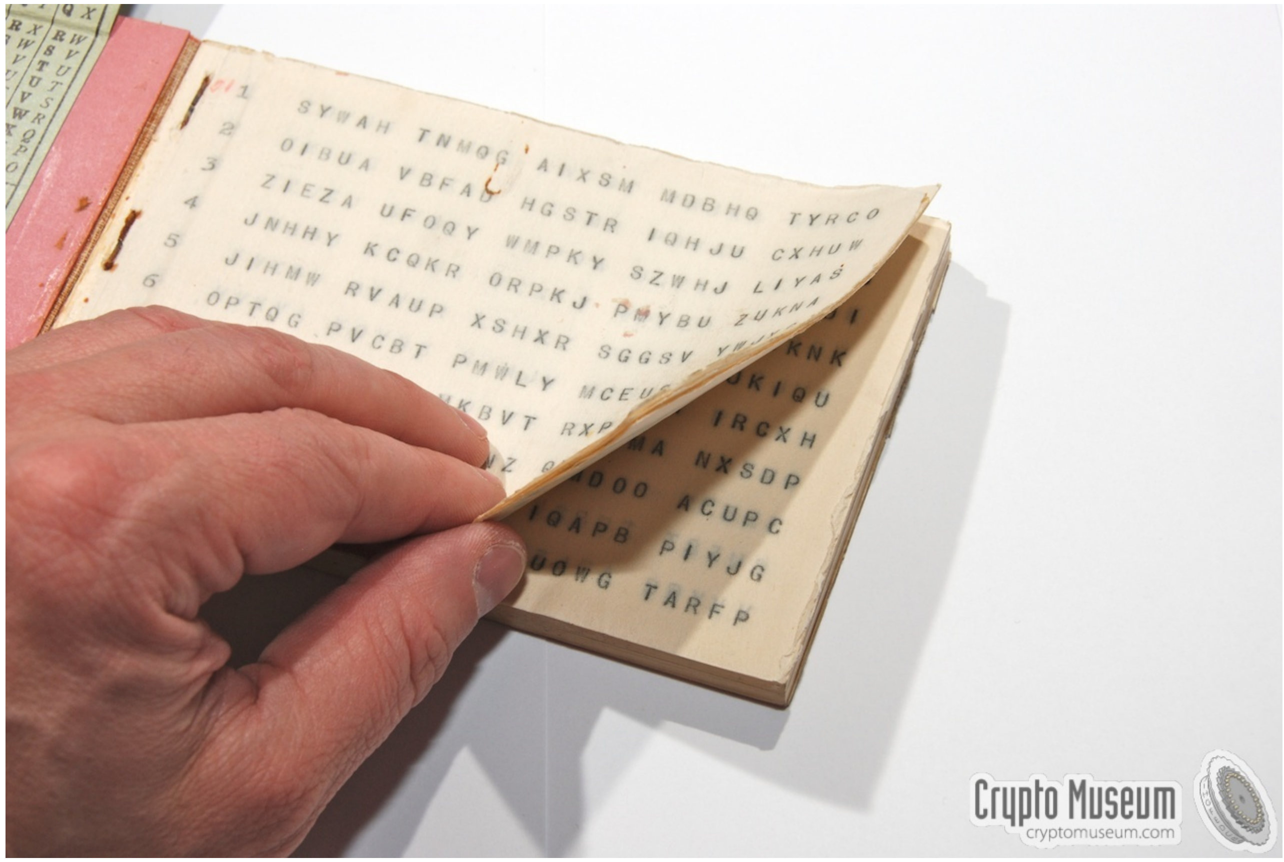
More Sophisticated Ciphers to Encrypt

- Advanced Encryption Standard or "AES" is a type of block cipher than uses a 128-bit key
- Used today in every https communication
- Security based on the assumption that no approach better than brute force is known (& bruce force takes too long)



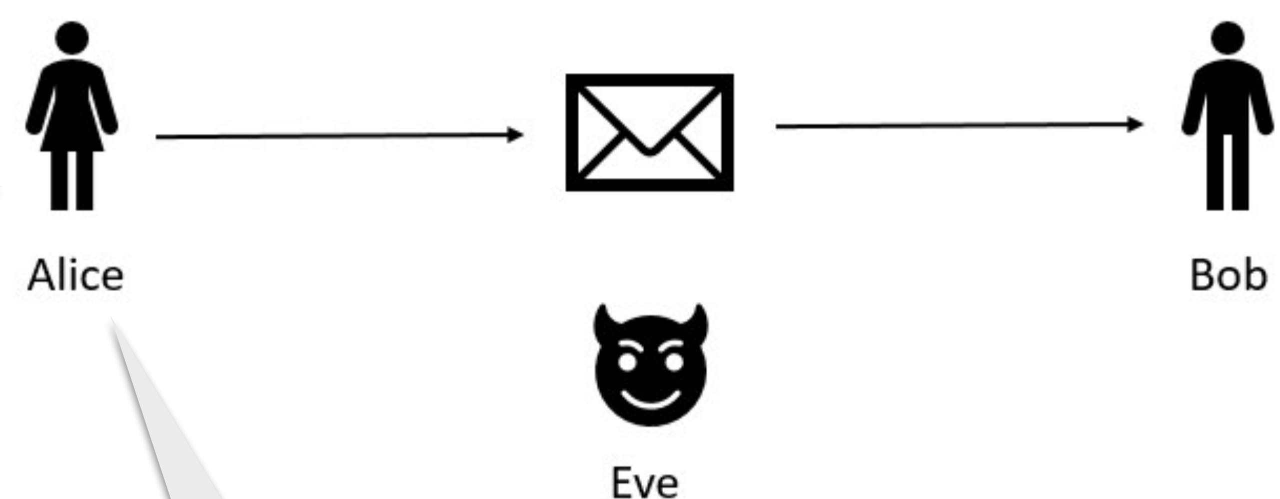
One-Time Pads

- Information theoretically secure
- XOR with random bits as long as the plaintext
- Only useful only for private-key cryptography (security)
    - Getting a key from Alice is just as hard as getting the original message



Public Key Cryptography

- Private-key cryptography requires securely sharing a private key
- Public-key cryptography: what if there is no secure channel to "preshare" a secret key?
    - Relies on the existence of one-way functions
    - Easy to compute, (cor



No pre-shared secret key between Alice and Bob

Each has a public key and a private key

One-Way Functions

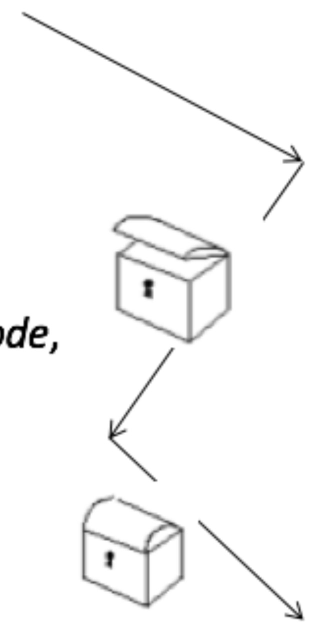
- A function $f(x) = y$ is one way iff
  - Given x, it is easy to compute $y = f(x)$
  - Given $y = f(x)$, it is hard to compute $x = f^{-1}(y)$
- E.g. Easy to compute 28487532223 * 72342452989 but hard to find factors of 2060857961121397335547
- Multiplication, Discrete logarithm are, probably, such functions (inverting them is not known to be in class **P**)
- Public-key crypto and secure encryption are based on the **assumption** that one-way functions exist

Why Care About Public Key Cryptography

HTTPS: SECURE INTERNET COMMUNICATION

Your browser makes an HTTPS request to the server, just like requesting any web page, plus its public key A.

The server responds by sending its public key B as a digital "trunk."



The browser makes up a *secret code*, K = a*B, and encodes its message using K

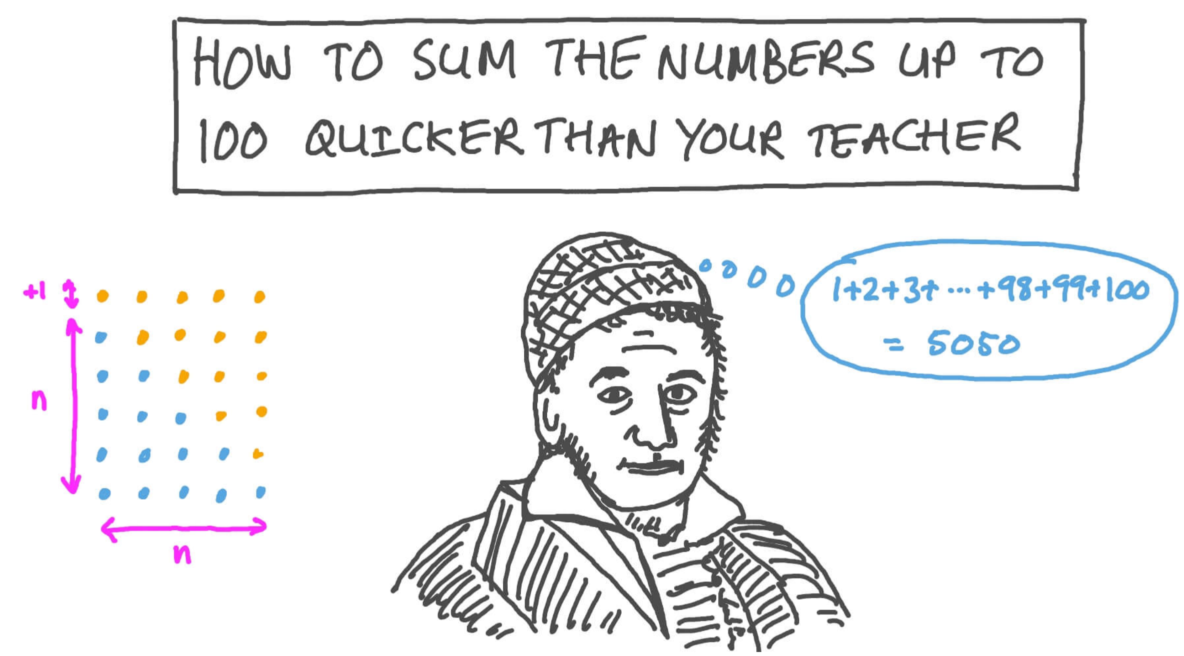
The server computes *secret code*, K = b*A, and decodes the message using K

From now on, the server and browser can now communicate in both directions using the secret code K, and no one else can read their conversation.

Back to Impagliazzo's Five Worlds

Story Setup: Grouse & Gauss

- Professor Grouse wants to humiliate Gauss in front of the class by inventing a problem that Gauss cannot solve
- In each of the five worlds, see whether Grouse wins or Gauss.
- Assume that Grouse and Gauss are polynomial-time algorithms
  - Gauss is faster than Grouse (as a mathematical genius)



Algorithmica (P = NP)

- *"Grouse cannot embarrass Gauss by giving him a problem that he can't solve (that Grouse can later demonstrate a solution of)"*
- All NP-complete problems become efficiently solvable!
- Challenges of AGI become trivial: can train the computer to perform any task humans can do
    - No way of telling computers & people apart (think Captchas)
    - Automate creativity
    - Computers can find proofs for any theorem!
- No security or cryptography, no way of making information available to some people without making it available to everyone

Heuristica

- NP problems are hard in the worst case, but easy on average
- Hard instances of problems exist, but they are also hard to find!
- *"Grouse might be able to find problems that Gauss cannot answer in class, but it might take Grouse a week to find a problem Gauss cannot solve in a month."*
- For practical purposes, indistinguishable from Algorithmica
- Heuristics for most NP-problems will work
- Still no way to ensure security or cryptography

Pessiland

- Problems are hard in the average case but no one-way functions
- Nothing good to say about this world
- Easy to generate hard instances of NP problems but hard to generate *solved* instances
  - A problem that no one knows the answer to not helpful for public-key cryptography
- *"Grouse could pose Gauss problems that even the budding genius could not solve. However, Grouse could not solve the problems either, and so Gauss's humiliation would be far from complete."*

Minicrypt

- One-way functions exist, so many private-key cryptography based security applications are possible:
  - Can digitally authenticate messages
  - If can preshare private key, can setup secure communication channels
- One-way functions can be used to generate hard *solved* problems
  - E.g. take x, and compute $y = f(x)$ where f is one-way
  - Pose the question, "Find any x' such that $f(x') = y$" knowing one solution x
- *Grouse finally gains the upper-hand and can best Gauss in front of the class*
- No public-key cryptography though

Cryptomania

- One-way functions exist and public-key cryptography is possible
- *"Gauss is utterly humiliated; by means of conversations in class, Grouse and his pet student would be able to jointly choose a problem that they would both know the answer to, but which Gauss could not solve. In fact, in such a world, Grouse could arrange that all the students except Gauss would be able to solve the problems asked in class."*
- Great for privacy, limits the capability of authorities to restrict privacy
- Closest to the real world, in that as far as we know, the RSA cryptosystem is secure
  - (Based on the assumption that factoring or discrete log are intractable problems)

Which World Do You Want to Live In?

[Brief]

Power of Randomness

Randomness: What is it Good For

- In many CS classes we have seen very efficient, very clean randomized solutions to computational problems
- Does randomization fundamentally change what class of problems we can *efficiently solve*?
  - First, randomization cannot fundamentally change whether a problem is decidable or not
  - Any TM that uses randomness can be simulated by one that is deterministic (by simulating all random choices)
  - Randomization is just an algorithmic tool for efficiency
- Does randomization **fundamentally** help us gain efficiency?

Randomness: What is it Good For

- Short answer: we don't think so

Long Answer: P versus BPP

- Class BPP (bounded-error probabilistic poly time): languages that can be solved by a probabilistic polynomial-time algorithm A

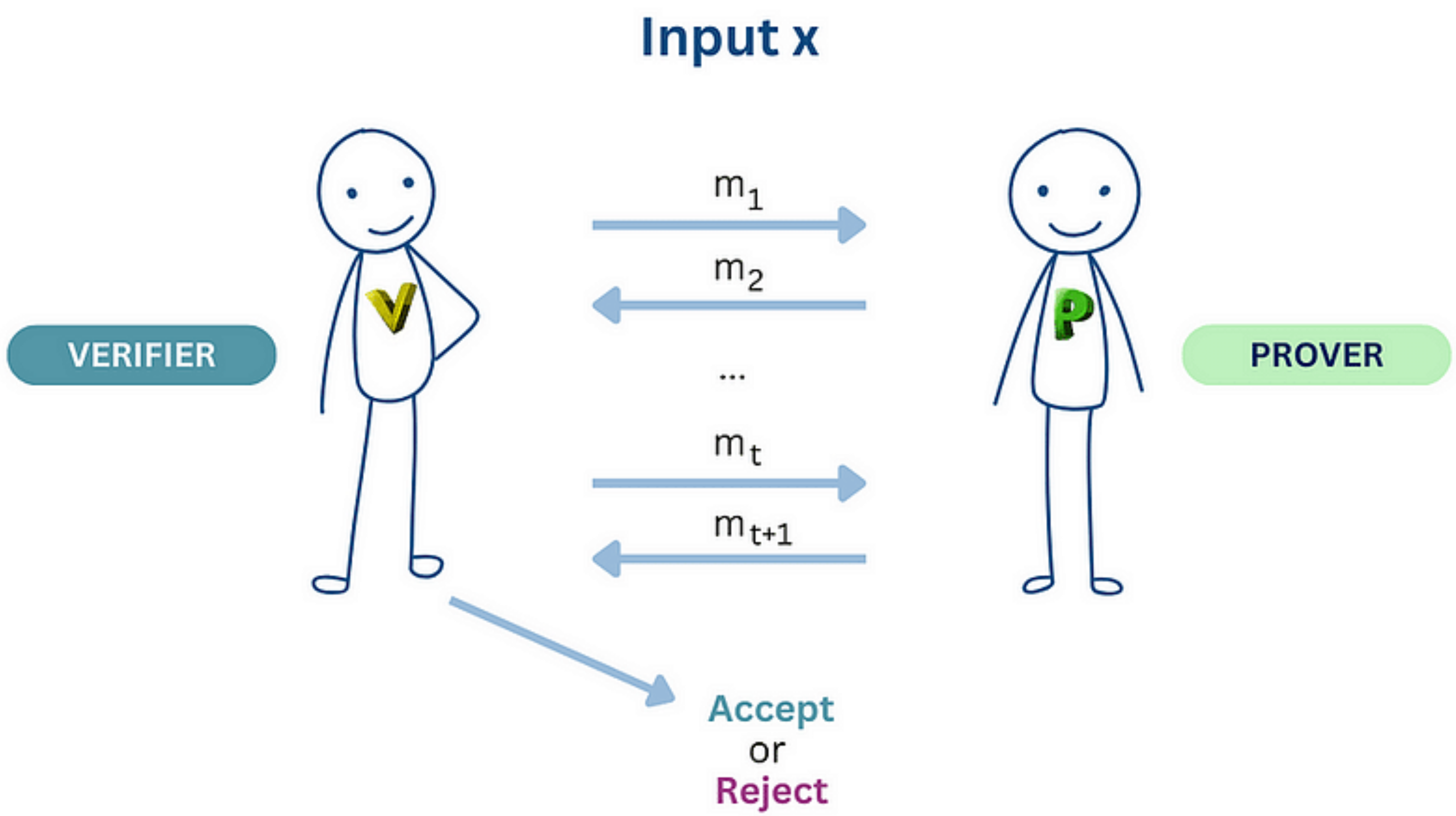
$$\text{If } x \in L, \text{ then } A \text{ accepts } x \text{ in } L \text{ with probability} \geq \tfrac{3}{4}.$$

$$\text{If } x \notin L, \text{ then } A \text{ accepts } x \text{ in } L \text{ with probability} \leq \tfrac{1}{4}.$$

- P versus BPP open problem:
  - Does every problem that has an efficient randomized algorithm also have an efficient deterministic one?
- Know that $\text{P} \subseteq \text{BPP} \subseteq \text{EXP}$, Current belief: $\text{P} = \text{BPP}$
  - *Haven't even been able to show much weaker* $\text{BPP} \subsetneq \text{NEXP}$

[Brief]

Power of Interaction in Verification



Class NP

- NP is the class of languages where given a "static" certificate (alleged proof), a polynomial-time verifier can quickly ascertain whether or not it is a valid certificate and accept/ reject
- NP $=$ languages with efficient verification using static proofs
  - **P $\neq$ NP** : assumption that verification is easier than solving from scratch

Verifying a Proof in Real Life

- Easier to verify an alleged proof if you ask questions
- **Question.** Are "interactive proofs" (proof systems where verifier can ask questions) fundamentally more powerful than static ones?
  - Yes, interaction adds quite a bit more power

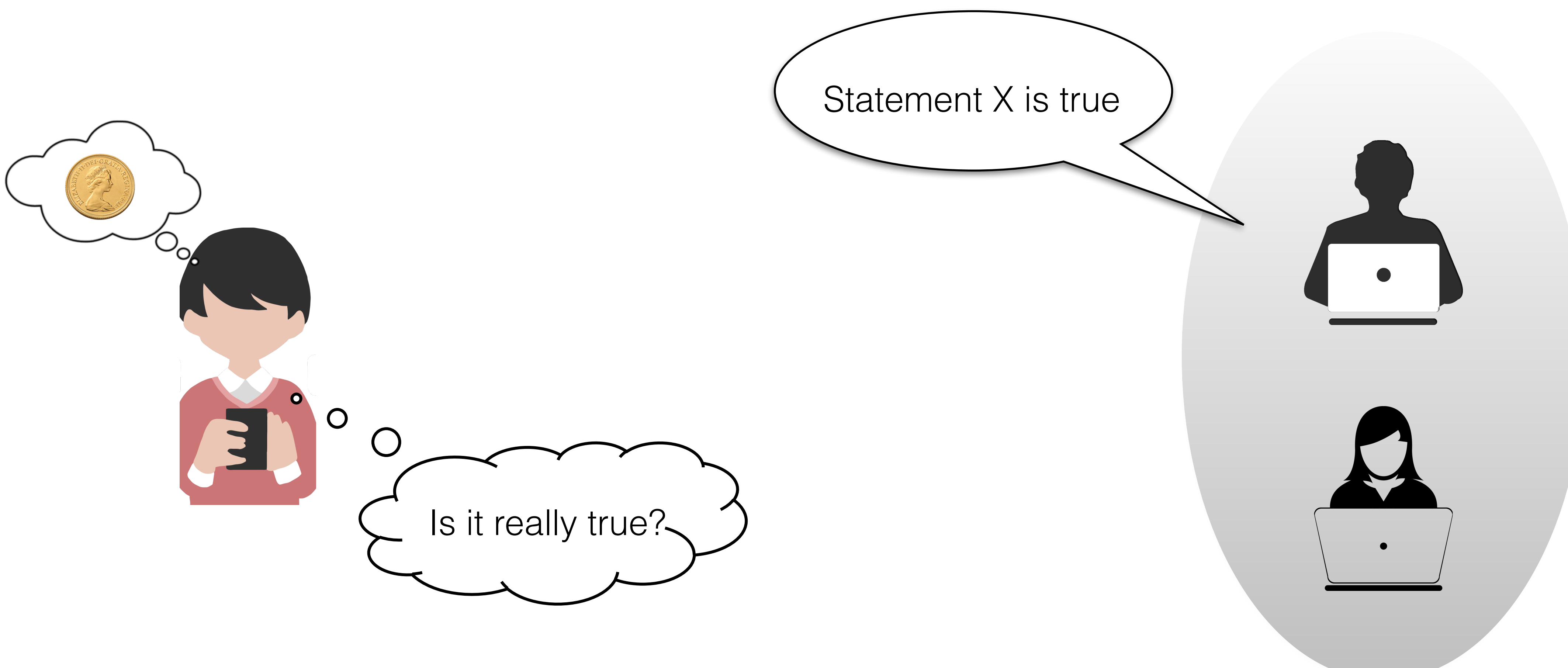
Verifying a Proof in Real Life

- Easier to verify an alleged proof if you ask questions
- **Question.** Are "interactive proofs" (proof systems where verifier can ask questions) fundamentally more powerful than static ones?
  - Yes, interaction adds quite a bit more power

Interactive Proofs (IP)

[GMR, BM 85, BGWW 88]

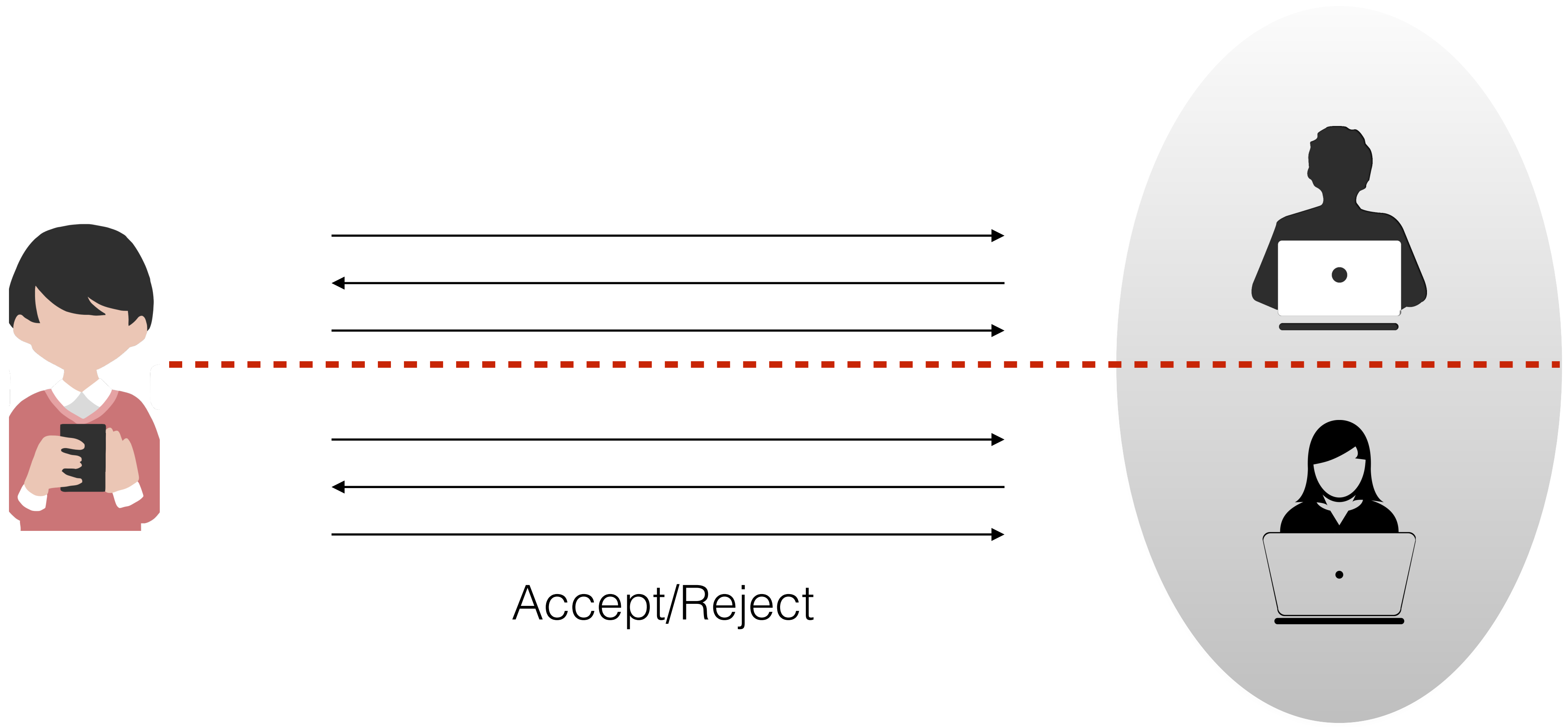
- Formal framework to study verification of outsourced computation
- Verifier is weak but can flip some coins, the provers are all-powerful
- Provers goal is to convince the verifier to accept their claim



Interactive Proofs (IP)

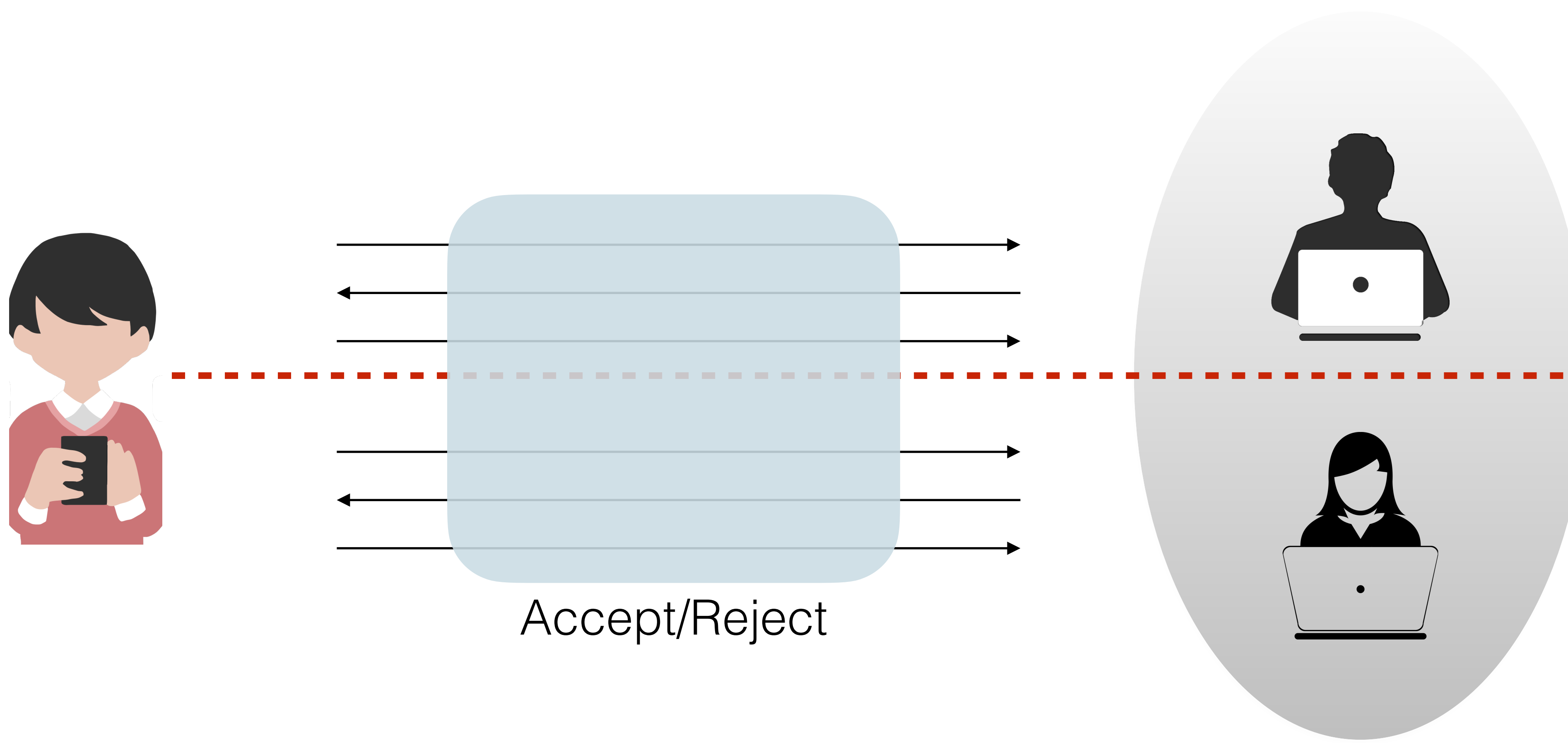
[GMR, BM 85, BGWW 88]

- Verifier interacts with each prover separately
    - Asking them questions to check if they are being truthful
- Finally, if Verifier is convinced, he accepts. Otherwise, he rejects



IP Guarantees

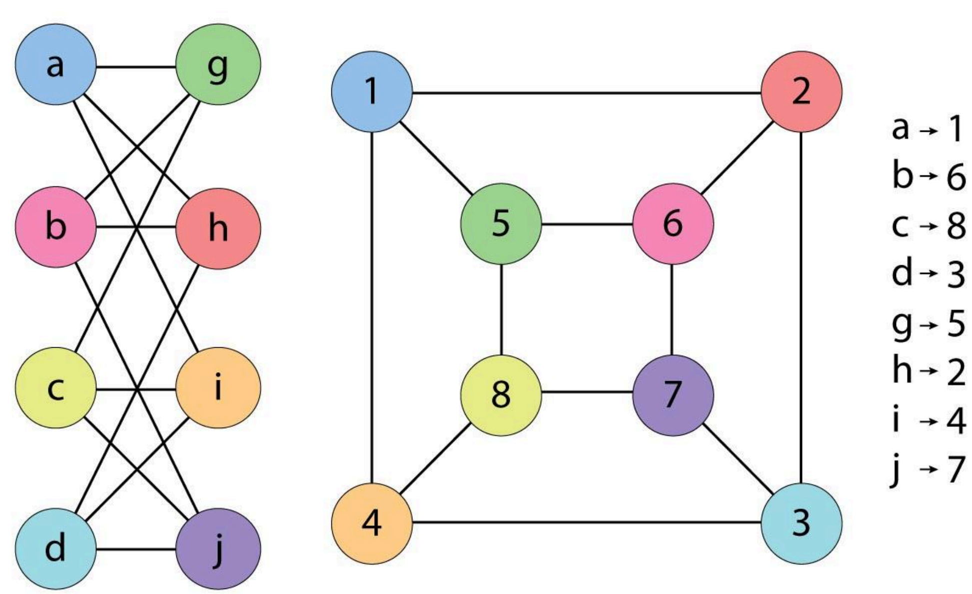
- Correctness: True statements should be provable
  - Verifier accepts them always (with prob 1)
- Soundness: False statements should NOT be provable
  - Verifier rejects them with most of the time (with prob ≥ 2/3)



Graph Non-Isomorphism

- Prove that two graphs are NOT isomorphic to each other

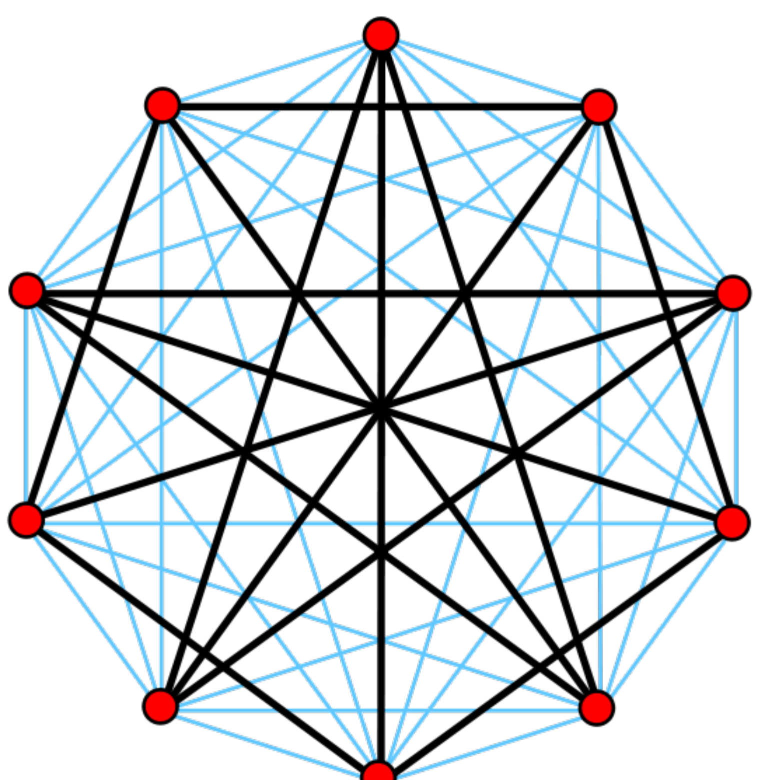
Definition: Graphs G and H are isomorphic (G ≄ H) if vertices of G can be relabeled to turn it into H. If no such labelling exists, they are non-isomorphic.

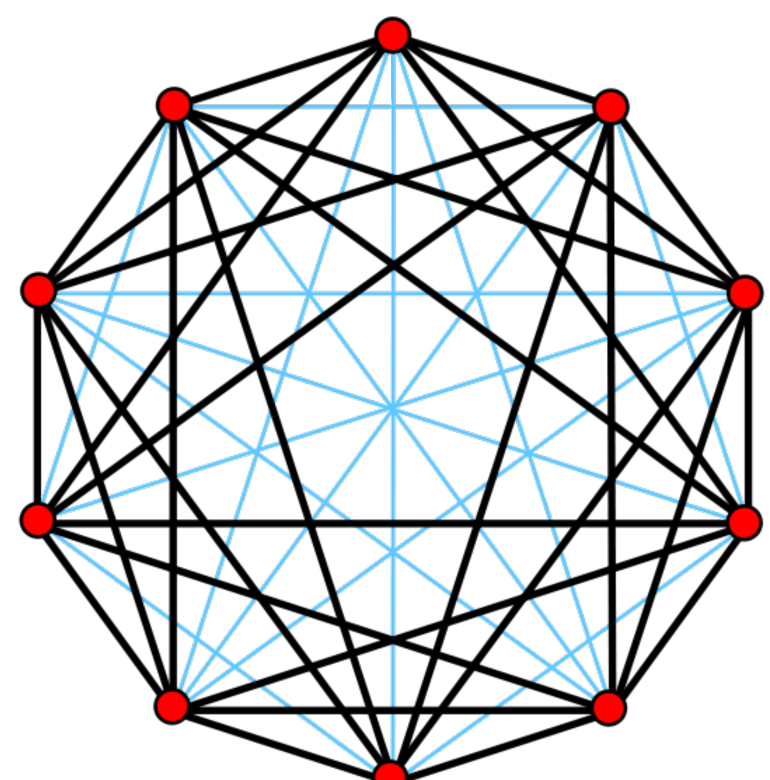


An example of graphs that ARE isomorphic

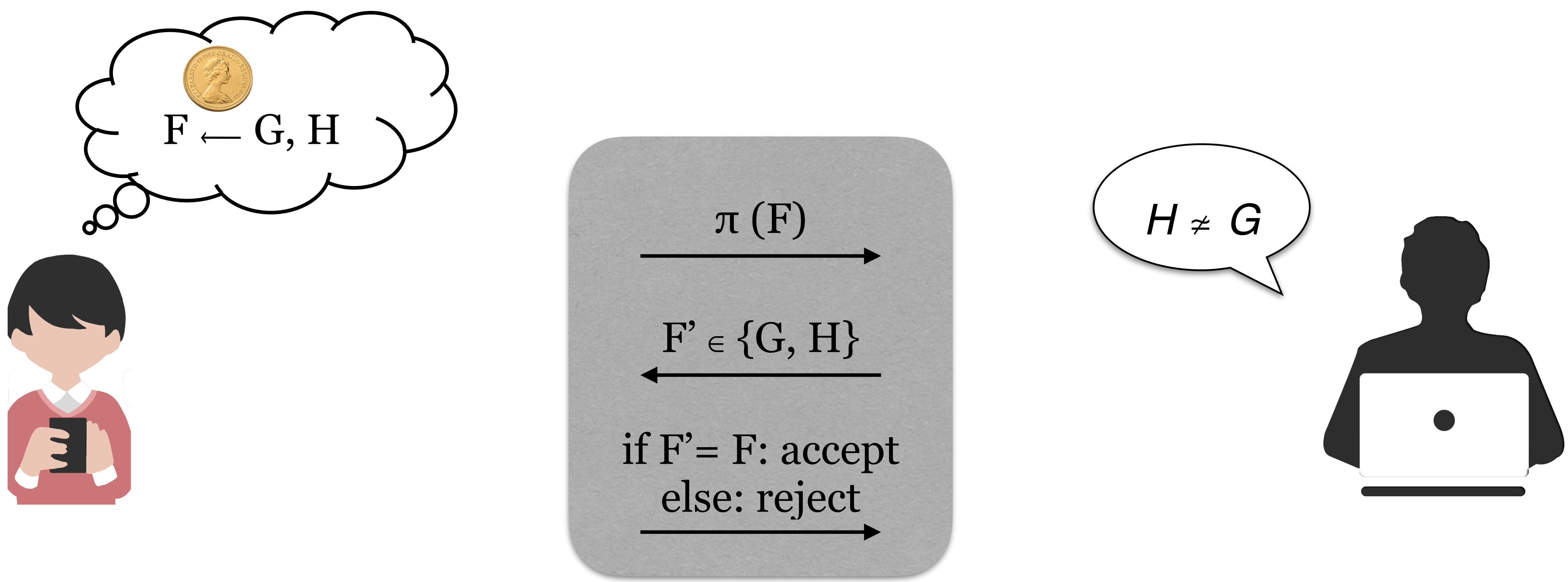
Proving Non-Isomorphism: Static Proof

- Naive approach: list all possible relabelings of G
- Check that none of them yield H
- Will need to check all n! relabeling (computationally infeasible)





IP for Graph Non-Isomorphism



- (Correctness) If H ≄ G: Prover can convince Verifier with Prob 1
- (Soundness) If H ≃ G: Prover is always caught with Prob at least 1/2

Rich History of IPs

- Widely-studied area with a rich history and deep results
- Most importantly, $\mathsf{IP} = \mathsf{PSPACE}$
- Also shown that any problem in NP admits extremely succinct interactive proofs (verified only needs to query $O(1)$ bits of the proof!)
  - Led to the area of PCPs [BFL90, BFLS91, AS92, FGLSS91, AS92, ALMSS92]
- Recently, widely-used as a framework to design efficient protocols for computation outsourcing
  - IP for Muggles [GKR08], Proofs of proximity [RVW13, GR15, KR15], etc.