CMSC 28100

Introduction to Complexity Theory

Spring 2024
Instructor: William Hoza

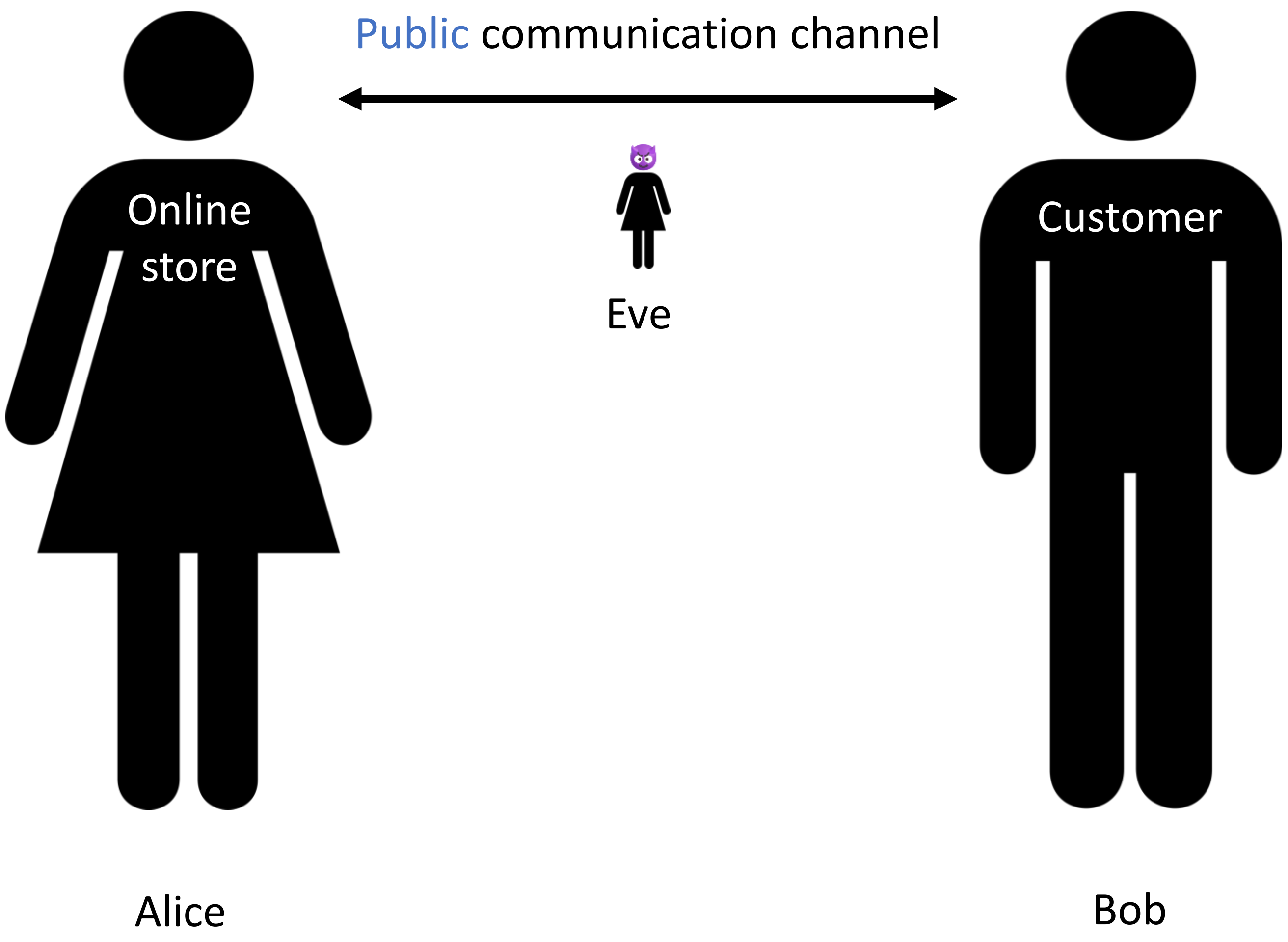
How to feel about intractability

- We have encountered several tractable problems in this course
  - DECOMPOSABLE-INTO-SQUARES, CIRCUIT-VALUE, 2-COLORABLE, …
  - Conventional attitude: This is “good news” 🙂
- We have also identified many problems that are probably/definitely intractable
  - HALT, BOUNDED-HALT, CIRCUIT-SAT, 3-SAT, CLIQUE, …
  - Conventional attitude: This is “bad news” 🙁
- Twist: Sometimes we are hoping that certain problems are intractable! 🙃

Cryptography

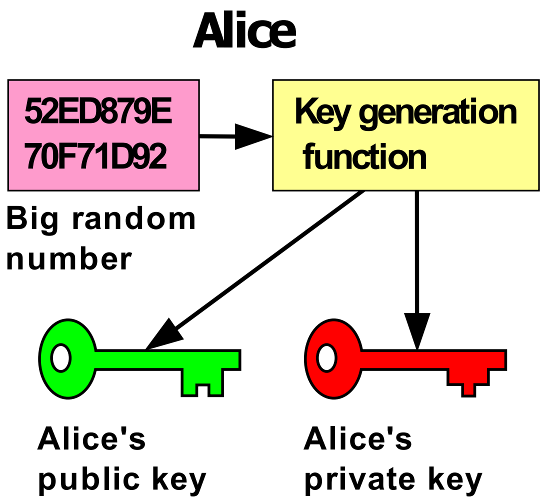
Secure communication

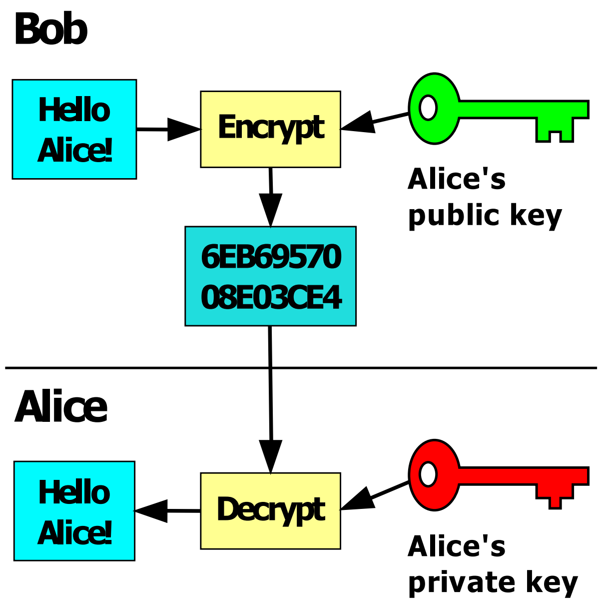
- How can Bob send a private message to Alice?
    - E.g., credit card number
- It seems impossible, because Alice and Eve receive all the same information from Bob!



- A clever approach: Try to force Eve to solve an intractable problem

Public-key encryption





- Alice’s advantage over Eve: Alice knows the private key and Eve doesn’t

Public-key encryption scheme

"keys" "encrypt" "decrypt"

- **Definition:** A simplified public-key encryption scheme is a triple (K, E, D), where:
  - $K \subseteq \{0,1\}^* \times \{0,1\}^*$ and $E, D: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$
  - For every $w \in \{0,1\}^*$ and every $(k_{\text{pub}}, k_{\text{priv}}) \in K$, we have $D\left(k_{\text{priv}}, E(k_{\text{pub}}, w)\right) = w$
  - E and D can be computed in polynomial time
  - For every $(k_{\text{pub}}, k_{\text{priv}}) \in K$, we have $|k_{\text{pub}}| = |k_{\text{priv}}|$

If Eve is computationally unbounded

- Let's show that if Eve has unlimited computational power, then encryption is futile

- **Claim:** There exists a function $D_{\text{Eve}}: \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that for every message $w \in \{0,1\}^*$ and every pair $(k_{\text{pub}}, k_{\text{priv}}) \in K$, we have

$$D_{\text{Eve}}\left(k_{\text{pub}}, E(k_{\text{pub}}, w)\right) = w$$

- **Proof:** If $E(k_{\text{pub}}, w) = E(k_{\text{pub}}, w') = y$, then $w = D(k_{\text{priv}}, y) = w'$ ✔

What if Eve is computationally bounded?

- Amazing fact: There are known public-key encryption schemes such that decrypting without the private key seems to be intractable!
  - (*Better: There are schemes such that it is apparently intractable to "occasionally" "partially" decrypt without the private key. Making this precise is beyond the scope of our course)
- Example: "RSA"
- These amazing encryption schemes make our modern internet experience possible! Can we prove that they are secure?

Cryptography and P vs. NP

- Let (K, E, D) be a simplified public-key encryption scheme
- There is a function D_{Eve} such that $D_{\mathrm{Eve}}\left(k_{\mathrm{pub}}, E(k_{\mathrm{pub}}, w)\right) = w$
- **Definition:** We say that (K, E, D) is horribly insecure if the function D_{Eve} can be computed in polynomial time

Theorem: If $\mathrm{P} = \mathrm{NP}$, then every simplified public-key encryption scheme is horribly insecure.

Cryptography and P vs. NP

Theorem: If $\mathrm{P} = \mathrm{NP}$, then every simplified public-key encryption scheme is horribly insecure.

- **Proof:** Let $L = \{\langle k_{\mathrm{pub}}, y, w\rangle : \text{there exists } z \text{ such that } E(k_{\mathrm{pub}}, wz) = y\}$
- $L \in \mathrm{NP}$: the witness is z. (Since D is poly-time-computable, z is poly-size)
- We are assuming $\mathrm{P} = \mathrm{NP}$, so therefore $L \in \mathrm{P}$
- Therefore, Eve can construct the message bit-by-bit in polynomial time

Cryptography and P vs. NP

- Disclaimer: The preceding discussion of public-key encryption is simplified
  - For example, a real encryption scheme should explain how to generate keys
- Nevertheless, the main message is accurate:
- If $\mathrm{P} = \mathrm{NP}$, then secure public-key encryption is impossible!

Cryptography and P vs. NP

- In fact, virtually all of theoretical cryptography relies on assumptions that are stronger than the assumption $\mathrm{P} \neq \mathrm{NP}$
- Maybe this makes you feel concerned about the uncertain foundations of computer security…
- Or, maybe this makes you feel more confident that $\mathrm{P} \neq \mathrm{NP}$, considering how much effort people expend trying to break cryptosystems

Coping with intractability

Facing intractability

- Suppose you need to solve some problem (for your job, your business, your hobby project, your research, …)
- You formulate your problem as a language L
- You find a proof (or some compelling evidence) that $L \notin \mathrm{P}$ 🙁
  - Undecidability, EXP-completeness, NP-completeness…
- What now? Is it time to give up?

Coping with intractability

- The fact that $L \notin \mathrm{P}$ does not necessarily mean that you cannot solve your problem
- There are, in fact, several approaches for coping with the fact that $L \notin \mathrm{P}$
- We will discuss a few approaches, without any proofs

Nontrivial exponential-time algorithms

- Even if L doesn't have a polynomial-time algorithm, it still might have a nontrivial algorithm

Theorem: There is an algorithm that determines whether a given n-variable 3-CNF formula is satisfiable in time $O(1.308^n)$.

- If your inputs happen to be relatively small, then maybe an exponential time complexity is tolerable

Structured inputs

- Another approach: Maybe you can identify some additional structure in the instances you care about, beyond the definition of L
- Example: Initially, you think you need to solve SAT

$$\mathrm{SAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable CNF formula}\}$$

- SAT is NP-complete 😳

Structured inputs

- However, after studying your situation more closely, you realize that your instances are all "Horn formulas"
- A Horn formula is a CNF formula with at most one positive literal per clause
- $\text{HORN-SAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable Horn formula}\}$
- Exercise: Prove that $\text{HORN-SAT} \in \text{P}$

SAT solvers

- Another approach: If your problem is in NP, you can try using a "SAT solver" (practical software for solving SAT)
- For example, many software package managers use SAT solvers to resolve dependencies
- Presumably, the reason this works is that there is some hidden structure in the SAT instances that come up in practice (think Horn formulas)

SAT solvers are not a panacea

- Note: The practical success of SAT solvers does not undermine the conjecture $\mathrm{P} \neq \mathrm{NP}$
- There are “hard instances” on which practical SAT solvers fail badly
- Cryptographers are very skilled at generating such instances!

Approximation algorithms

- The next approach that we will discuss for coping with intractability is approximation algorithms
- This approach only makes sense if you are trying to solve an optimization problem
- Example: the Knapsack problem

The Knapsack problem

- Given: Positive integers $w_1, \ldots, w_k, v_1, \ldots, v_k, B$
  - Interpretation: There are k items
  - Item i has weight w_i (in pounds) and value v_i (in dollars)
  - We can carry up to B pounds of stuff in our knapsack
- Goal: Find a set $S \subseteq \{1, 2, \ldots, k\}$ such that $\sum_{i \in S} v_i$ is as large as possible, subject to the constraint $\sum_{i \in S} w_i \leq B$

KNAPSACK is NP-complete

$$\text{KNAPSACK} = \{\langle w_1, \dots, w_k, v_1, \dots, v_k, B, V\rangle : \text{there exists } S \subseteq \{1, 2, \dots, k\}$$
$$\text{such that } \Sigma_{i \in S}\, w_i \leq B \text{ and } \Sigma_{i \in S}\, v_i \geq V\}$$

Theorem: KNAPSACK is NP-complete

Approximation algorithms for Knapsack

- For every $w_1, \dots, w_k, v_1, \dots, v_k, B$, define

$$\mathrm{OPT} = \max\left\{\sum_{i \in S} v_i : S \subseteq \{1, \dots, k\} \text{ and } \sum_{i \in S} w_i \le B\right\}$$

Theorem: There exists a poly-time algorithm such that given $w_1, \dots, w_k, v_1, \dots, v_k, B$, the algorithm outputs $S \subseteq \{1, \dots, k\}$ such that:

- $\sum_{i \in S} w_i \le B$
- $\sum_{i \in S} v_i \ge 0.99 \cdot \mathrm{OPT}$

Approximation algorithms for Knapsack

- For every $w_1, \ldots, w_k, v_1, \ldots, v_k, B$, define

$$\mathrm{OPT} = \max\left\{\sum_{i \in S} v_i : S \subseteq \{1, \ldots, k\} \text{ and } \sum_{i \in S} w_i \leq B\right\}$$

Theorem: For every $\epsilon > 0$, there exists a poly-time algorithm such that given $w_1, \ldots, w_k, v_1, \ldots, v_k, B$, the algorithm outputs $S \subseteq \{1, \ldots, k\}$ such that:

- $\sum_{i \in S} w_i \leq B$
- $\sum_{i \in S} v_i \geq (1 - \epsilon) \cdot \mathrm{OPT}$

Approximation algorithms are not a panacea

- In some cases, approximation algorithms take some of the sting out of NP-completeness
- However, in other cases, approximation algorithms are unhelpful!

Inapproximability of the clique problem

- For a graph G, let $\omega(G)$ be the size of the largest clique in G

Theorem: Suppose there exists a poly-time algorithm such that given a graph $G = (V, E)$, the algorithm outputs a clique $S \subseteq V$ satisfying $|S| \geq 0.01 \cdot \omega(G)$. Then $\mathrm{P} = \mathrm{NP}$.

Inapproximability of the clique problem

- For a graph G, let $\omega(G)$ be the size of the largest clique in G

Theorem: Let $\epsilon > 0$, and suppose there exists a poly-time algorithm such that given a graph $G = (V, E)$, the algorithm outputs a clique $S \subseteq V$ satisfying $|S| \geq \epsilon \cdot \omega(G)$. Then $\mathrm{P} = \mathrm{NP}$.

Quantum computing

- Another approach for coping with intractability: Quantum Computing
- A quantum computer is a computational device that uses special features of quantum physics
- A detailed discussion of quantum computing is outside the scope of this course
- We will discuss only some key facts about quantum computing

Quantum computing

- Quantum computers are, to some extent, hypothetical
- So far, researchers have constructed rudimentary quantum computers
- There are huge ongoing efforts to build fully-functional quantum computers

Quantum complexity theory

- One can define a complexity class, BQP, consisting of all languages that could be decided in polynomial time by a fully-functional quantum computer
- The mathematical definition of BQP is beyond the scope of this course
- One can prove that $\mathrm{BPP} \subseteq \mathrm{BQP} \subseteq \mathrm{PSPACE}$

Shor's algorithm

- Recall $\mathrm{FACTOR} = \{\langle N, K\rangle : N \text{ has a prime factor } p \leq K\}$
- **Conjecture:** $\mathrm{FACTOR} \notin \mathrm{P}$

Theorem (Shor's algorithm): $\mathrm{FACTOR} \in \mathrm{BQP}$

- FACTOR is a likely counterexample to the extended Church-Turing thesis!