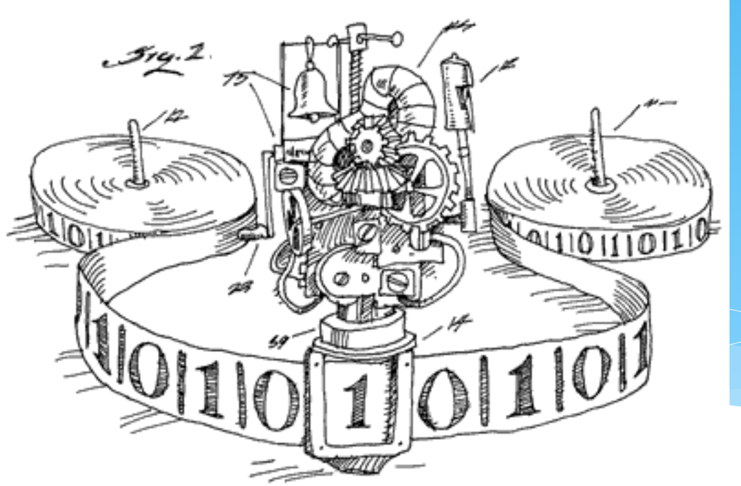
EECS 376: Foundations of Computer Science

Chris Peikert

5 April 2023



Cryptography

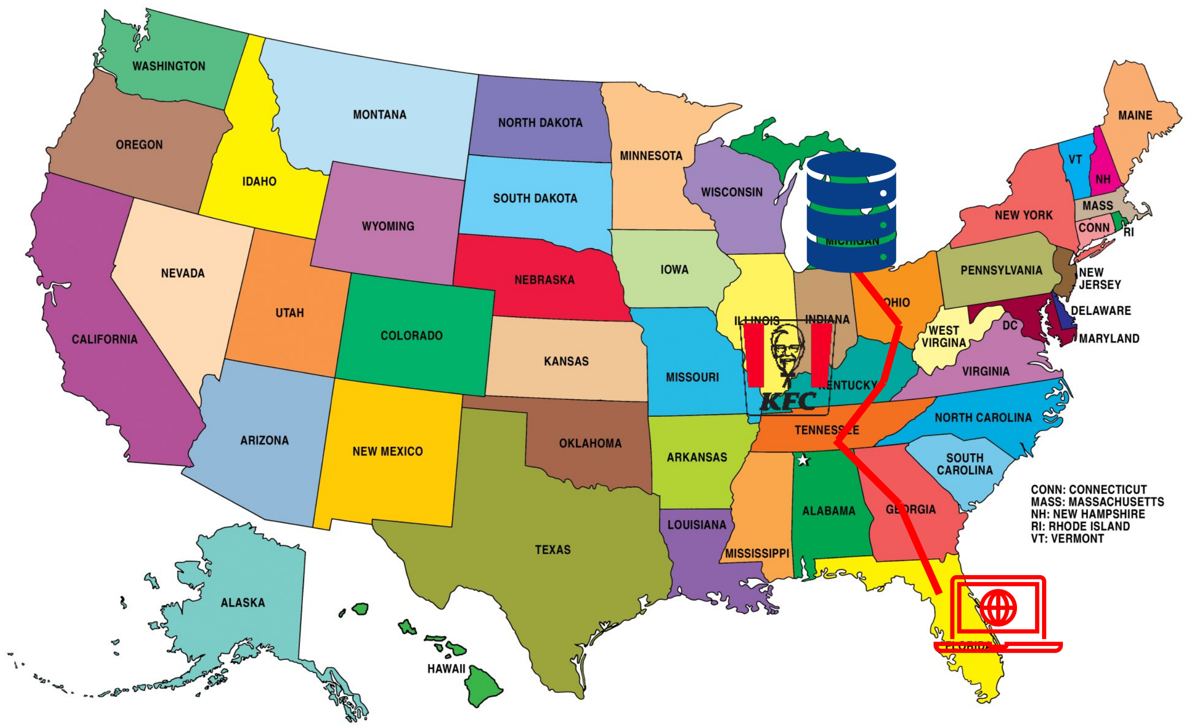
Cryptography's Core Goals

1) **Confidentiality:** ensuring that only *intended recipients* can read data. (Tool: **encryption**.)
2) **Integrity:** ensuring that data has not been *undetectably altered*. (Tool: **hashing**.)
3) **Authenticity/authentication:** ensuring that data *came from a claimed source* / that an entity is *who it claims to be*. (Tool: **signatures/MACs**.)

WARNING!!:
The crypto we are about to show you in insecure in many ways. Do not use!
Take EECS 388/475/575 for more info.

Confidentiality: Motivation

- Communication on the internet is “public”: data sent over it may be *intercepted* en route to its destination.



Q: Can we prevent an *eavesdropper* from learning the messages we send?

Model: Alice, Bob, and Eve

- Two parties **Alice** and **Bob** communicate over a *<u>public channel</u>*, and there is an eavesdropper **Eve** that *<u>sees all the data they send</u>*.

- **Ideally:** Eve should not be able to “understand” their messages, *<u>even if she knows their communication protocol</u>*.
- **Example:** “ig-pay atin-lay” is easily decoded if one knows the protocol.

Kerckhoffs's principle: *“A cryptosystem should be secure even if everything about the system, **except the key**, is public knowledge.”*

Cryptosystem Security

* **Kerckhoffs's principle:** *"A cryptosystem should be secure even if everything about the system,* ***except the key****, is public knowledge."*

* Two types of security:

1. **Information-Theoretic (unconditional):** Eve cannot break security, even using unbounded computation.
2. **Computational (conditional):** In order to break security, Eve will have to solve a (*conjectured*) computationally hard problem.

One-Time Pad Encryption

- Beforehand, Alice and Bob agree on a ***uniformly random secret key*** (a string over the message alphabet).
- Alice ***encrypts*** her message (of the same length as the key) by "padding" it with ("adding" it to) the secret key.
- Bob ***decrypts*** the message by subtracting the secret key.

secret key

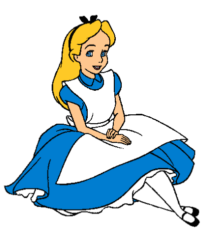
V	N	O	O	P

plaintext

	H	E	L	L	O
+	V	N	O	O	P
	C	R	Z	Z	B

ciphertext

	C	R	Z	Z	B
-	V	N	O	O	P
	H	E	L	L	O



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

One-Time Pad Details

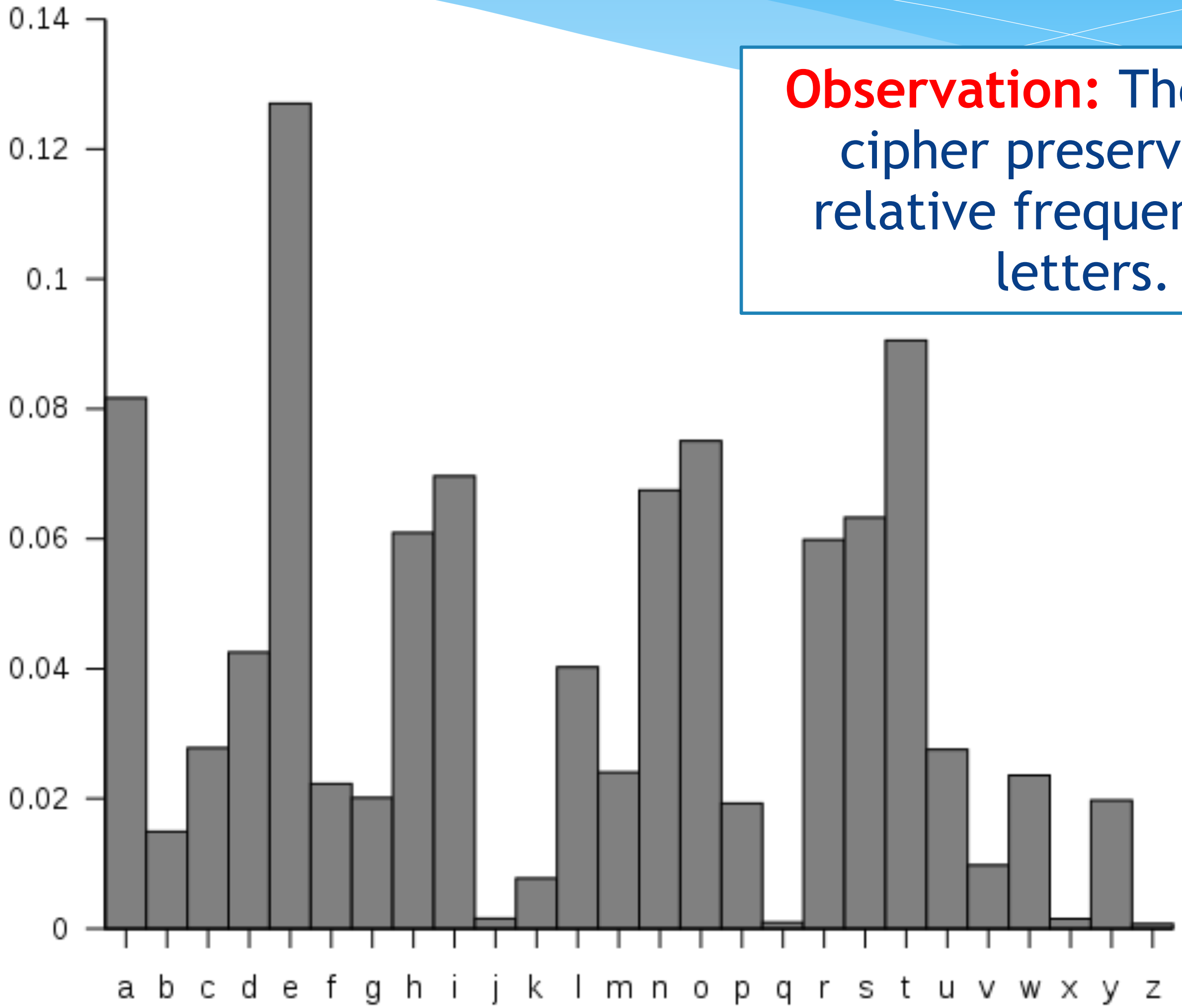
* Let $m = m_1 m_2 \ldots m_n$ be a message and $k = k_1 k_2 \ldots k_n$ be a secret key.
* m_i and k_i are bits, or alphabet chars (e.g., from $\{0,1,\ldots,25\}$)
* **Encryption $E_k(m)$:** $c_1 \equiv m_1 + k_1 \pmod{26}$, $c_2 \equiv m_2 + k_2 \pmod{26}$, …
* **Decryption $D_k(c)$:** $m_1 \equiv c_1 - k_1 \pmod{26}$, $m_2 \equiv c_2 - k_2 \pmod{26}$, …
* **Information-Theoretic Security:** From c, Eve "learns nothing" about m that she didn't already know beforehand!
* **Downside 1:** The key must be as long as the message.
* **Downside 2:** It's insecure to use the same key twice.
* **Downside 3:** Alice and Bob must agree on a ***secret*, *uniformly random key*** beforehand.

Smaller Key: Caesar Cipher

* **Encryption $E_k(m)$:** $c_1 \equiv m_1 + k_1 \pmod{26}$, $c_2 \equiv m_2 + k_2 \pmod{26}$
* **Downside 1:** n key symbols k_i are required
* **Idea:** reuse some key symbol(s)?
* **Caesar cipher:** for all i: $k_i = s$ (just *one* random symbol)
* **Encryption $E_s(m)$:** for all i: $c_i \equiv m_i + s \pmod{26}$
* **Observation:** The *frequency* of symbols in c match those of m.
* **Conclusion:** easily breakable by "frequency analysis":
  e.g., E (4) , T (19) are the most common in English text.
* **Exercise:** Find m and s, given $E_s(m) = \text{CADCQ}$.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Relative Frequencies of Letters in the English Language

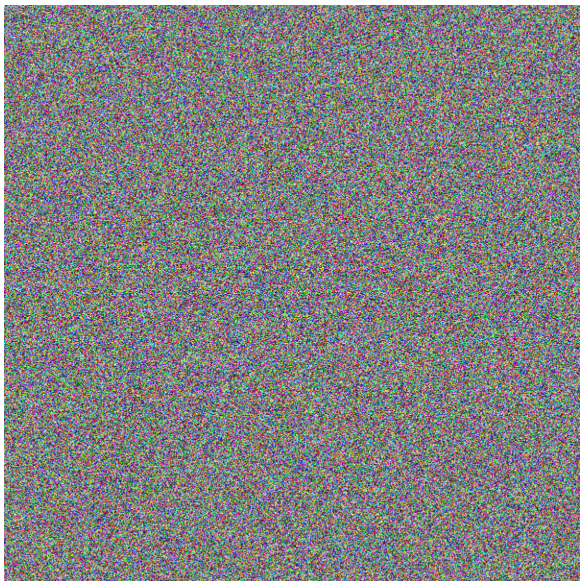


Observation: The Caesar cipher preserves the relative frequencies of letters.

Insecurity of Two-Time Pad

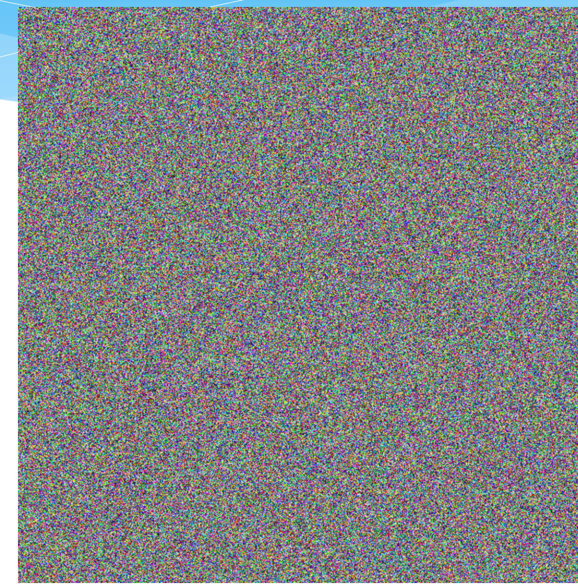
* Let $m = m_1 m_2 \ldots m_n$ be a message.
* Let $k = k_1 k_2 \ldots k_n$ be a secret key.
* **Encryption $E_k(m)$:** $c_1 \equiv m_1 + k_1 \pmod{26}$, $c_2 \equiv m_2 + k_2 \pmod{26}$
* **Downside 2:** Can't use the same key twice.
* Let $m' = m_1' m_2' \ldots m_n'$ be a (different) message.
* **Encryption $E_k(m')$:** $c_1' \equiv m_1' + k_1 \pmod{26}$, $c_2' \equiv m_2' + k_2 \pmod{26}$
* **Consider:** $d_1 \equiv c_1' - c_1 \equiv m_1' + k_1 - \left(m_1 + k_1\right) \equiv m_1' - m_1 \pmod{26}$
* **Observation:** $m' - m$ is not random!
* **Conclusion:** Can use statistical attacks

Two-Time Pad Attack



Ciphertext $c = m + k$

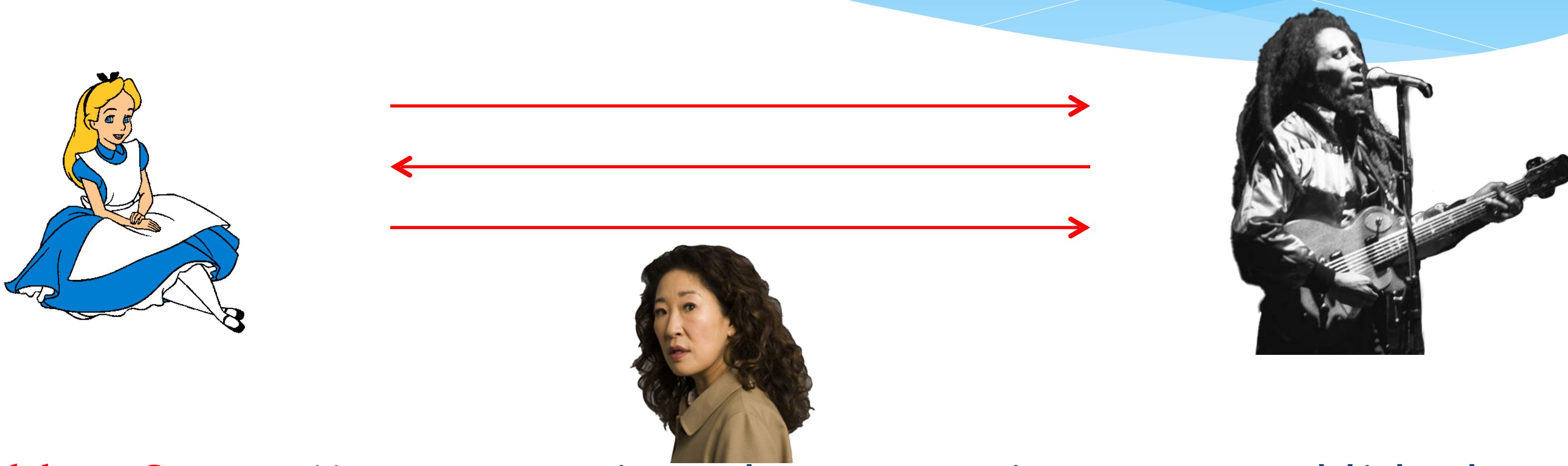
$c - c' = m - m'$



Ciphertext $c' = m' + k$

- **Key point:** Plaintexts are not uniformly distributed!
  - Same core weakness in Caesar cipher, substitution schemes, and reusing one-time pads: ciphertexts are *correlated* in exploitable ways.

Establishing a Shared Secret Key



Problem Setup: Many encryption schemes require a pre-established secret key. So Alice wants to a establish a secret with Bob—but Eve sees all their communication.

Question: (How) can two entities set up a shared *secret* key over a *public* channel?

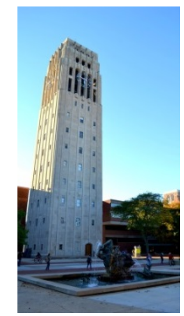
A Tale of Two Towers

- The Emperor of the North Tower wants to send a gift  to the Emperor of the Central Tower.
- The emperors never leave their towers.
- Their couriers travel back and forth, but they steal anything that is unlocked.
- If the box has one Emperor's lock, the other Emperor cannot open it.
- **Question:** Can the gift be sent securely?

North Tower

Central Tower

A Tale of Two Towers



* Emperor from the North Tower adds her lock
* Emperor from the Central Tower adds his lock
* Emperor from the North Tower removes her lock
* Emperor from the Central Tower removes his lock

Review: Number Theory

* Two integers a and b are ***equivalent modulo*** an integer $n \geq 2$, denoted $\boldsymbol{a \equiv b \pmod{n}}$, if they have the same remainder when divided by n.
* **Fact:** In modular addition/multiplication, we can replace any number with an equivalent one:

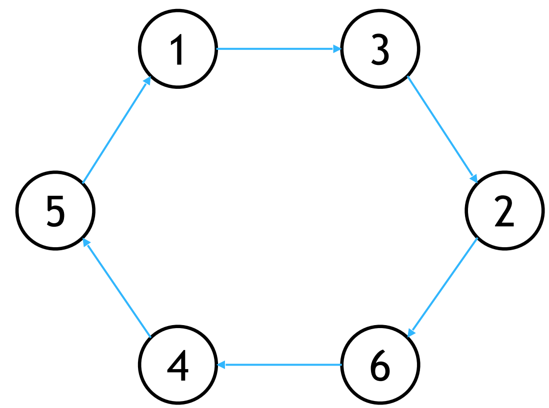
$$37 + 42 \equiv 1 + 0 \equiv 1 \pmod 3$$

$$1024 \cdot 152 \equiv 4 \cdot 2 \equiv 3 \pmod 5$$

* ***Fast modular exponentiation:***
$$3^8 \equiv (9)^4 \equiv 4^4 \equiv (16)^2 \equiv 1^2 \equiv 1 \pmod 5$$

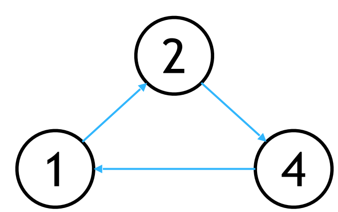
A Mathematical "Lock"

* Let p be a prime and let $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$.
* An integer g is a ***generator*** of $\mathbb{Z}_p^*$ if, for every $x \in \mathbb{Z}_p^*$, there exists $i \in \mathbb{N}$ such that $g^i \equiv x \pmod{p}$.
* **Example:** 3 is a generator of $\mathbb{Z}_7^*$, but 2 isn't.
* **Fact:** $\mathbb{Z}_p^*$ has a generator for *any* prime p.

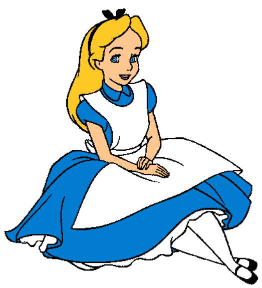


Discrete Log Conjecture: Given (large) prime p, generator g of $\mathbb{Z}_p^*$, and $x \in \mathbb{Z}_p^*$, there is no *efficient* algorithm for finding $i \in \mathbb{N}$ such that $g^i \equiv x \pmod{p}$.

Probably an "$\mathbf{NP}$-*Intermediate*" problem.



Diffie-Hellman Protocol



$x = (g^{a} \bmod p)$ $\longrightarrow$

$\longleftarrow$ $y = (g^{b} \bmod p)$

System parameters: a huge prime p and a generator g of $\mathbb{Z}_p^*$

Alice chooses *secret*, *random* $a \in \mathbb{Z}_p^*$, sends $x = (g^a \bmod p)$ to Bob.

Bob chooses *secret*, *random* $b \in \mathbb{Z}_p^*$, sends $y = (g^b \bmod p)$ to Alice.

Their secret shared key is $k = \left(g^{ab} \bmod p\right)$.

$$\left.\begin{array}{l}\text{Alice } \underline{\textit{locally}} \text{ computes: } y^a \equiv \left(g^b\right)^a \equiv g^{ba} \pmod{p}. \\ \text{Bob } \underline{\textit{locally}} \text{ computes: } x^b \equiv \left(g^a\right)^b \equiv g^{ab} \pmod{p}.\end{array}\right\}$$

Key: These are equal!

Diffie-Hellman: Example

Secret information is **bold, in red.**

- **Toy Example:** Alice and Bob use published modulus $p = 23$ and generator $g = 5$ of $\mathbb{Z}_p^*$.
  - Alice chooses secret random $\boldsymbol{a} = \mathbf{4}$, sends Bob $x = g^{\boldsymbol{a}} \bmod p = 5^{\mathbf{4}} \bmod 23 = 4$.
  - Bob chooses secret random $\boldsymbol{b} = \mathbf{3}$, sends Alice $y = g^{\boldsymbol{b}} \bmod p = 5^{\mathbf{3}} \bmod 23 = 10$.
  - Alice computes $\boldsymbol{k} = y^{a} \bmod p = 10^{4} \bmod 23 = \mathbf{18}$
  - Bob computes $\boldsymbol{k} = x^{b} \bmod p = 4^{3} \bmod 23 = \mathbf{18}$
  - Alice and Bob now share a secret key!: $\mathbf{18}$
- **Observation:** Alice and Bob compute the same value $x^{\boldsymbol{b}} \bmod p = g^{\boldsymbol{ab}} \bmod p = g^{\boldsymbol{ba}} \bmod p = y^{\boldsymbol{a}} \bmod p$.

Diffie-Hellman: Security

- Eve sees p, g, $x = g^a \bmod p$, and $y = g^b \bmod p$.
- Eve wants to compute $k = g^{ab} \bmod p$.
- **DH Assumption:** There is no *efficient* algorithm that given g, p, $(g^a \bmod p)$, and $(g^b \bmod p)$ finds $(g^{ab} \bmod p)$.
- **Best *known* attack:** solve DLog to find a (or b).
- **Upshot:** Hard problems are sometimes a *good* thing!
- Most modern cryptographic protocols have ***conditional*** security guarantees: secure if there one-way functions exist, $\mathbf{P} \neq \mathbf{NP}$, DH/RSA/lattices are hard, etc…