Cryptography

Techniques and Paradigms in this Course

- Divide-and-conquer, greed, dynamic programming, the power of randomness

  Problems that are **easy** for a computer

- Computability

  Problems that are **impossible** for a computer

- NP-completeness and approximation algorithms

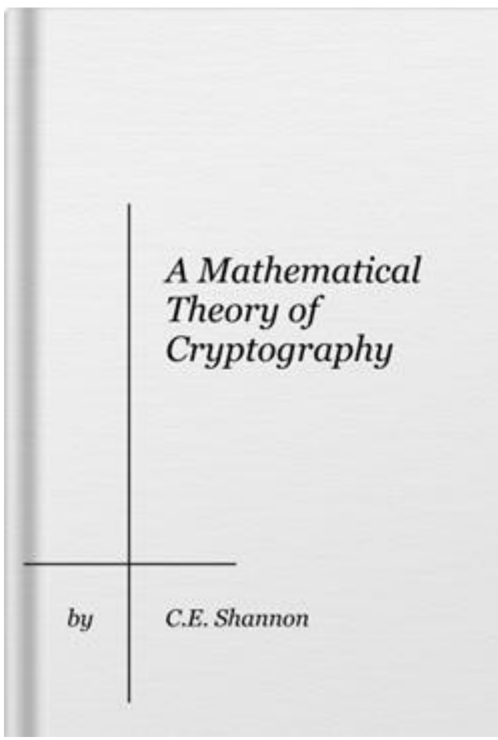
  Problems that are **"probably hard"** for a computer

- Cryptography

  Using "probably hard" problems for our benefit (hiding secrets)

The “probably hard” problems for cryptography are likely **not** NP-hard problems…we’ll see why in a bit

First “cryptography”: Egyptian tomb from from 20th century BCE debatably had a substitution cipher



First mathematical theory of cryptography: Claude Shannon (1945)

Cryptography's Core Goals

Build "cryptosystems" to achieve:

1) **Confidentiality:** ensuring that only *intended recipients* can read data. (today + next time)

2) **Integrity:** ensuring that data has not been *undetectably altered*. (last time)

3) **Authenticity:** ensuring that data came from a particular entity. (next time)

WARNING!

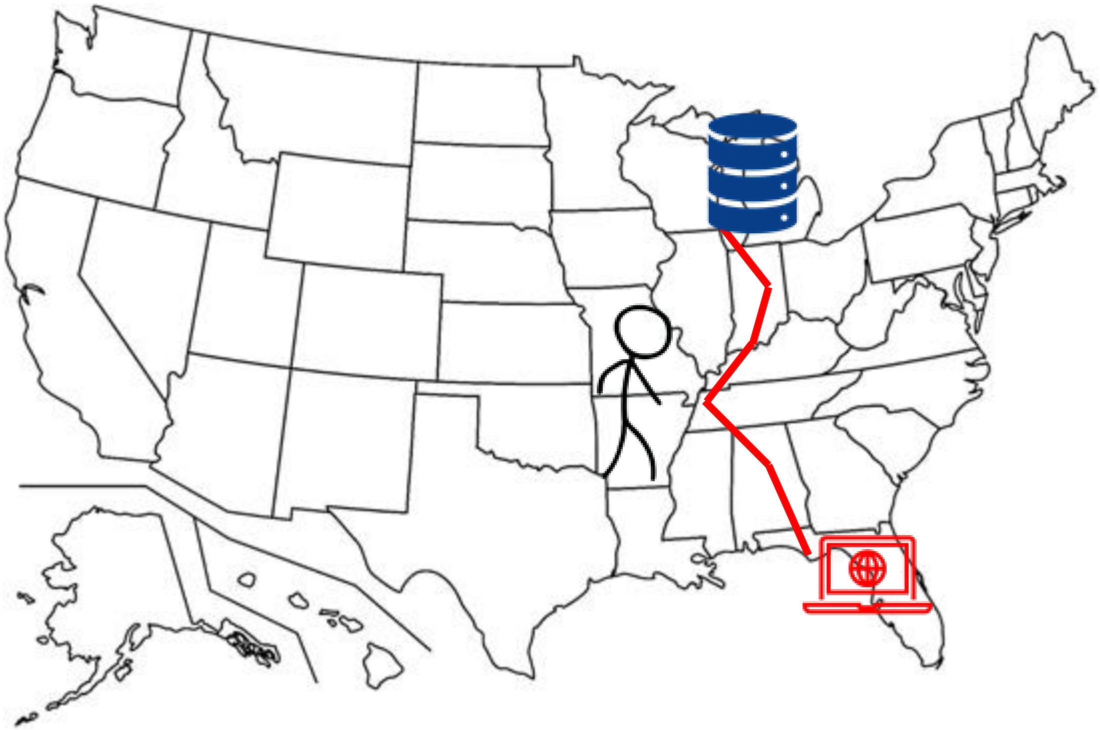
The crypto we are about to show you in insecure in many ways. Do not use!

Take EECS 388/475/575 for more info.

Communication on the internet is "public"

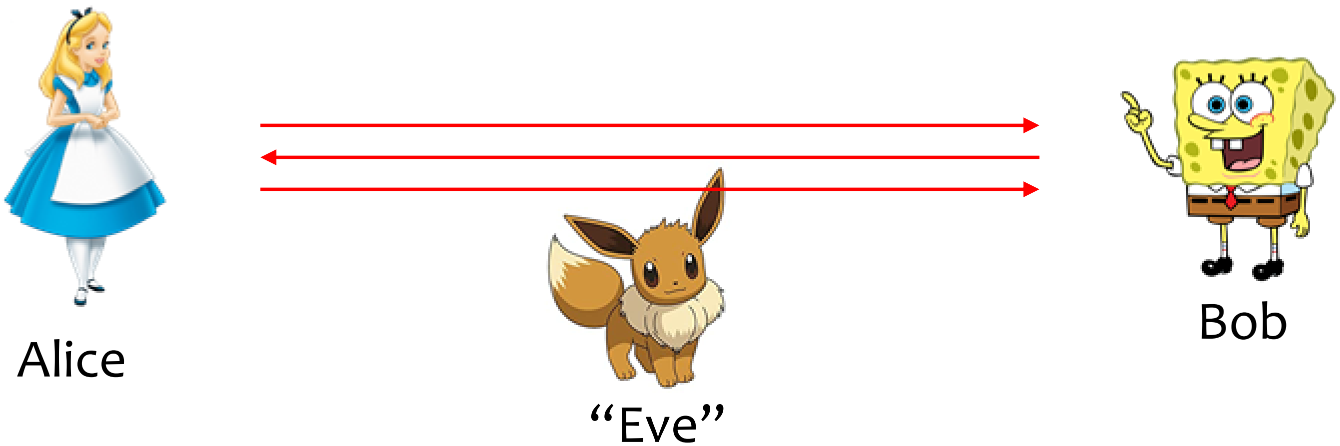
Data may be *intercepted* en route to its destination.

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



Model: Alice, Bob, and Eve

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



Ideally: Eve can’t decode Alice and Bob’s messages, *even knowing their protocol*.

e.g. “ig-pay atin-lay” is easily decoded if you know the protocol.

Two Types of Security

This one is better, but
often unattainable

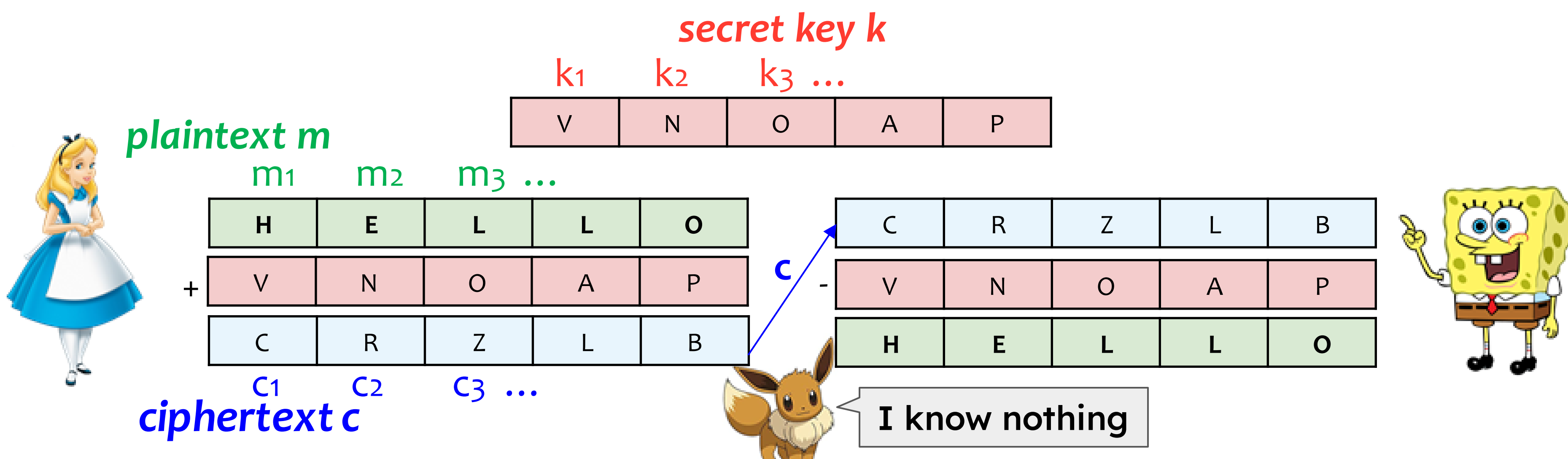
1. **Information-Theoretic (unconditional):** Eve cannot learn anything about their messages, even using unbounded computation.

2. **Computational (conditional):** In order to learn anything about the messages, Eve would have to solve a (*conjectured*) computationally hard problem.

First we'll see a scheme for achieving information-theoretic security
but with some other drawbacks…

One-Time Pad Encryption

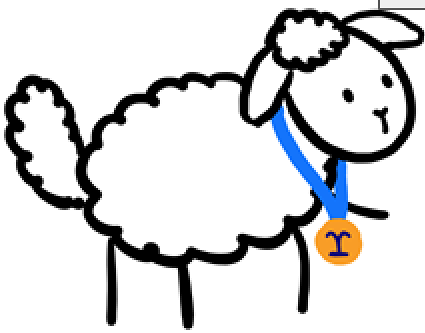
- Beforehand, Alice and Bob agree on a **uniformly random secret key k** = $k_1, k_2, k_3, \ldots$.
- Alice *encrypts* her **message m** (of the same length as the key) by *adding* it to the **secret key k**.
  - **ciphertext c**: $c_i = m_i + k_i \pmod{26}$
- Bob *decrypts* by *subtracting*: $m_i = c_i - k_i \pmod{26}$



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 Encryption

Ok great! Now that we have a shared secret key, here are my messages



secret key k

k_1	k_2	k_3 ...		
V	N	O	A	P

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

	C	**E**	**L**	**L**	**O**
+	V	N	O	A	P
	Q	R	Z	L	B

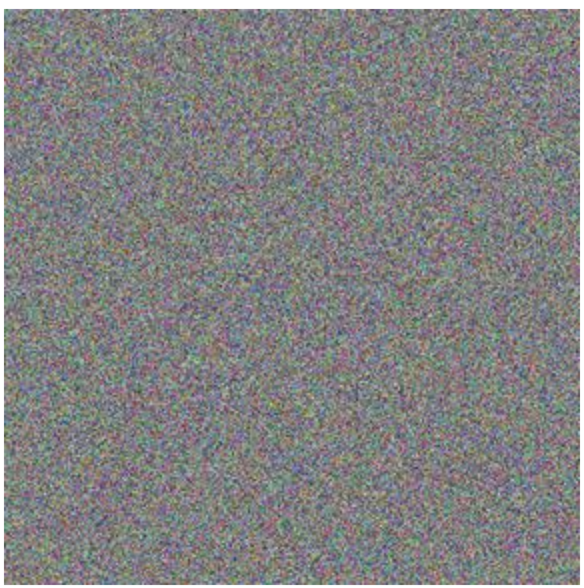
There's a reason it's called "one-time"

One-Time Pad Encryption

In general, using the same key twice allows Eve to figure out the *difference* between the messages, which violates security.

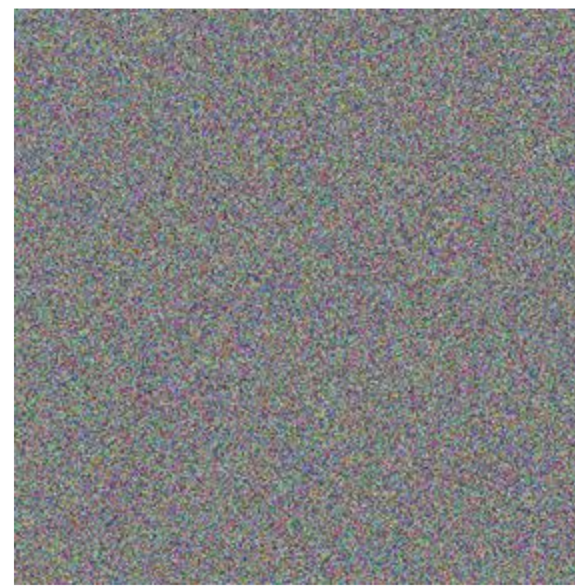
$$c = m + k \pmod{26} \qquad c' = m' + k \pmod{26}$$

$$c - c' \pmod{26} = m - m' \pmod{26}$$



c

m - m'



c'

The Caesar Cipher

An extreme example of using the same key multiple times

Using the same length-1 key k for every character

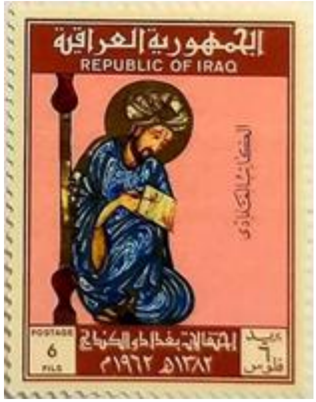
Add k to each letter of the message:

ciphertext c: $c_i = m_i + k \pmod{26}$

Observation: Caesar cipher preserves relative frequencies of letters.

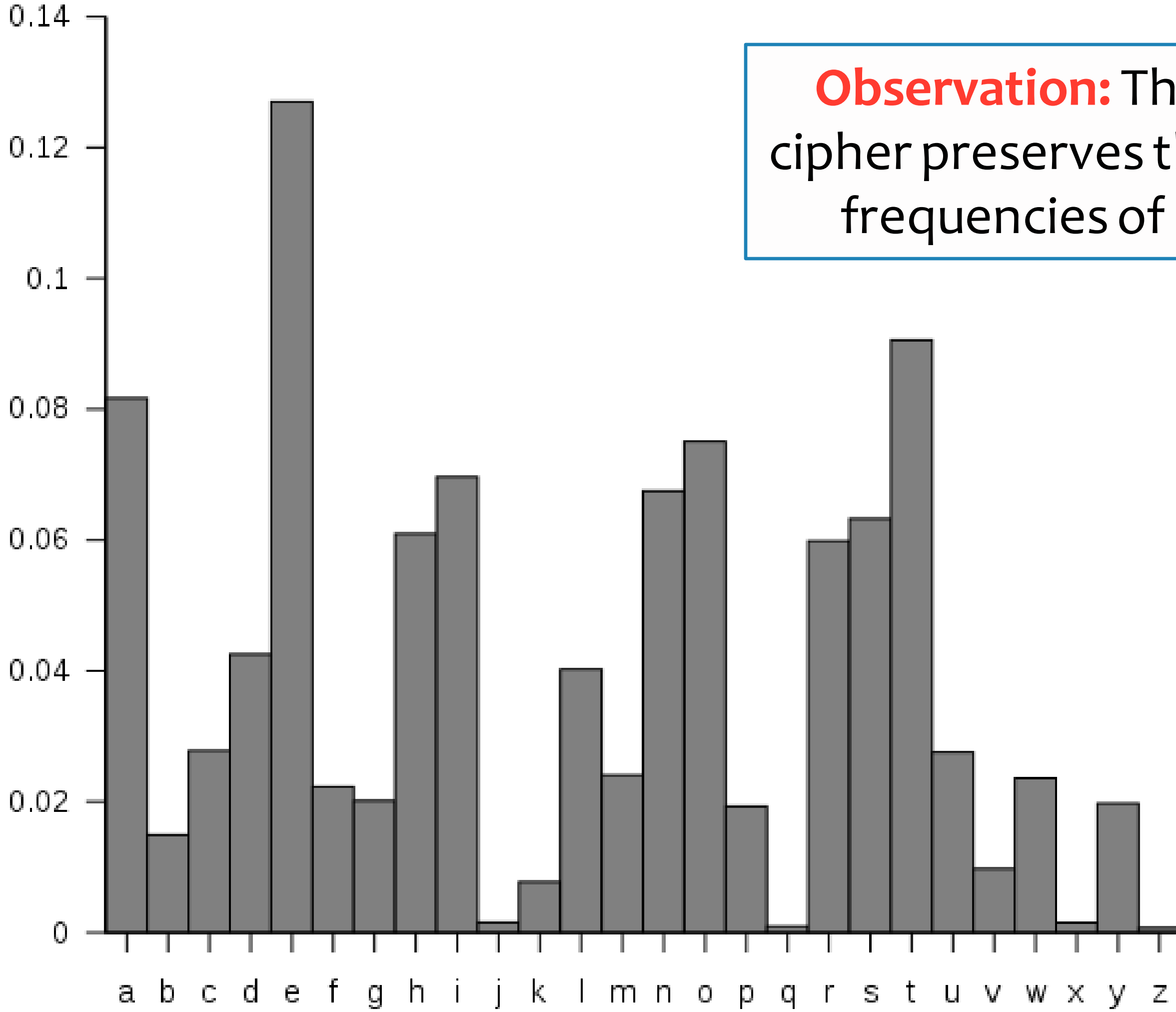
Conclusion: Caesar cipher is easily breakable by "frequency analysis"

e.g., E and T are most common in English text.



al-kindi
(~800)

Relative Frequencies of Letters in the English Language



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

One-Time Pad Pros and Cons

- **Pro: Information-Theoretic Security:** Eve "learns nothing" about the message, without knowing the secret key.
- **Con 1:** It's insecure to use the same key twice.
- **Con 2:** Alice and Bob must privately agree on the secret key beforehand.

It turns out these cons are necessary: To achieve information-theoretic security, you need to share in advance a random secret at least as large as the message.

So let's aim for **computational security** instead.

Today we will address **Con 2**.

Can Alice and Bob efficiently establish a secret over a public channel?

It seems impossible… but it's actually possible!

Can Alice and Bob efficiently establish a secret over a public channel?

Today: Diffie-Hellman Public Key Exchange (1976)

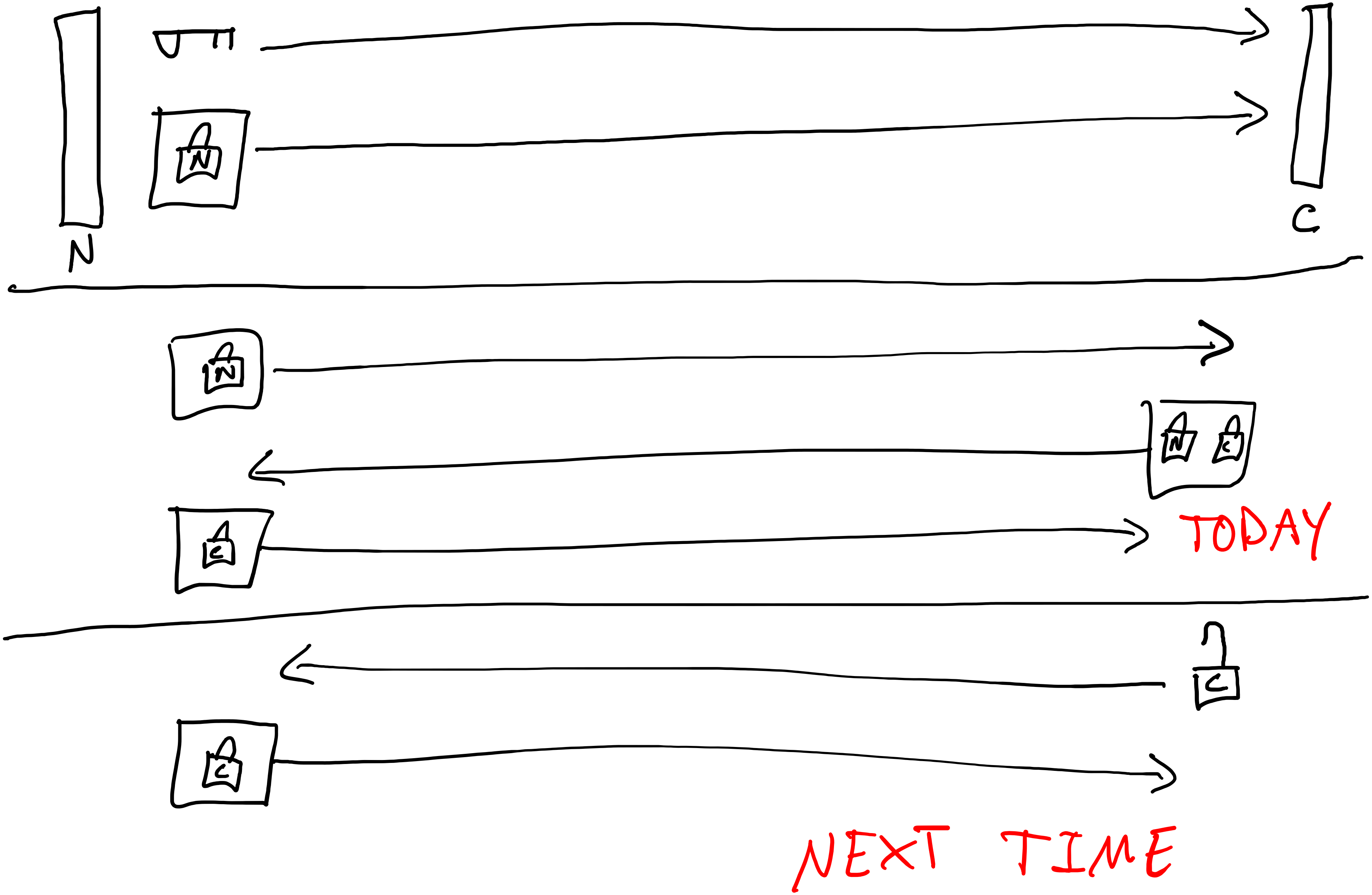
But first, a thought experiment:

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.
- Messengers can travel back and forth (multiple times), but they steal the gift if the box is unlocked. (They don't steal anything else.)
- Each Emperor has a lock and a key to their lock, but 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


N
C
TODAY
NEXT TIME

“Hard Problems” for Cryptography

Computational security: In order to break security, Eve would have to solve a (*conjectured*) computationally **hard problem**.

We would like to use an **NP-complete** problem as our hard problem, but there’s an issue:

We need our hard problem to be hard for **random inputs**, not just worst-case input.

We don’t have any conjectured hard-on-average **NP-complete** problems!

BUT we do have conjectured hard-on-average **number theory** problems!

*Researchers think these number theory problems are in a complexity class called NP-intermediate: In NP, but not in P nor NP-complete

Number Theory Review

- 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 a number with another 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:***

Repeatedly square the base and halve the exponent:

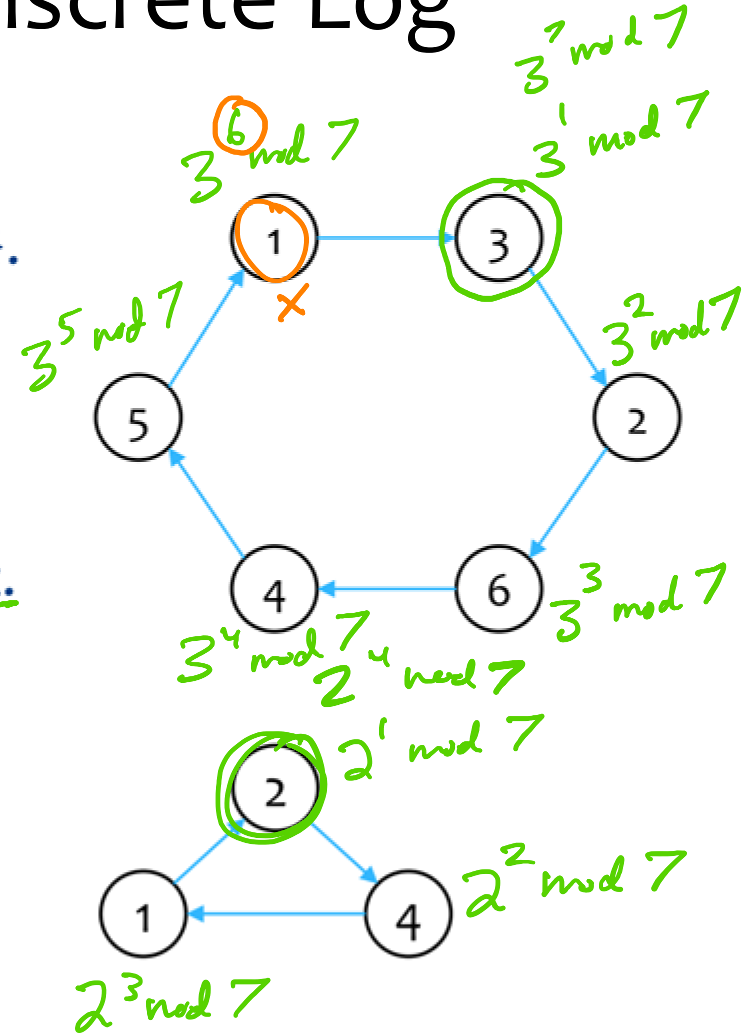
$$3^{10} \equiv (3^2)^5 \equiv 9^5 \xrightarrow{\text{mod } 5} 4^5 \equiv 4 \cdot 4^4 \equiv (4^2)^2 \equiv 4 \cdot 16^2 \xrightarrow{\text{mod } 5} 4 \cdot 1^2 \equiv 4 \pmod{5}$$

$a^b \pmod{5}$: brute force : b multiplications (exponential)

fast: $O(\log b)$ multiplications

A “Hard Problem”: Discrete Log

- 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 all prime p.



> **Discrete Log Conjecture:** Given 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 **NP**-*Intermediate* problem.

randomly chosen

i $\log_g x$

*It’s known that if discrete log is hard for worst-case input, it’s hard for random input

Now back to this:

Can Alice and Bob efficiently establish a secret over a public channel?

It seems impossible… but it's actually possible!

Diffie-Hellman Key Exchange

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

$p=23,\ g=5,\ a=4,\ b=3$

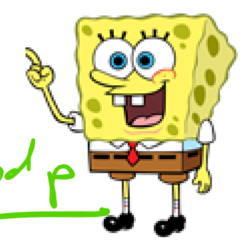
I pick a random
$a \in \mathbb{Z}_p^*$
and compute
$x = (g^a \bmod p)$

How?

$x = (g^a \bmod p)$

I pick a random
$b \in \mathbb{Z}_p^*$
and compute
$y = (g^b \bmod p)$

$y = (g^b \bmod p)$



$y^a \bmod p = \underline{g^{ba} \bmod p}$

$x^b \bmod p = \underline{g^{ab} \bmod p}$

Claim: Now Alice and Bob share the secret $\underline{k = (g^{ab} \bmod p)}$

Why can Alice and Bob both compute k?

Diffie-Hellman Key Exchange

Secret information is in **bold red**, public information is in blue

- **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 $\boldsymbol{k} = y^{\boldsymbol{a}} \bmod p = 10^4 \bmod 23 = \mathbf{18}$
  - Bob computes $\boldsymbol{k} = x^{\boldsymbol{b}} \bmod p = 4^3 \bmod 23 = \mathbf{18}$
  - Alice and Bob now share a secret key!: $\mathbf{18}$

Diffie-Hellman Security

Why can't Eve figure out ***k***?

Well… because we assume that.

DH Assumption: There is no *efficient* algorithm that given $\underbrace{g,\ p,\ (g^a \bmod p), \text{ and } (g^b \bmod p)}_{\text{what Eve knows}}$ finds $\underbrace{(g^{ab} \bmod p)}_{k}$.

Best known attack: Solve the **discrete log** problem to compute a from g^a mod p (or compute b from g^b mod p).

Discrete Log Conjecture: Given 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 $g^i \equiv x \pmod{p}$.

randomly chosen

Probably an **NP**-*Intermediate* problem.

What can Alice and Bob do once they have a shared secret ***k***?

Use ***k*** as a one-time pad!

Cryptography and Quantum Computers

Quantum computers can solve discrete log in polynomial time!

[Shor, 1994]

We don't have large-scale quantum computers (yet).

"Post-quantum" cryptography is an active area of research!

(Chris Peikert's bonus lecture)

Computing Modular Multiplicative Inverse in Polynomial Time

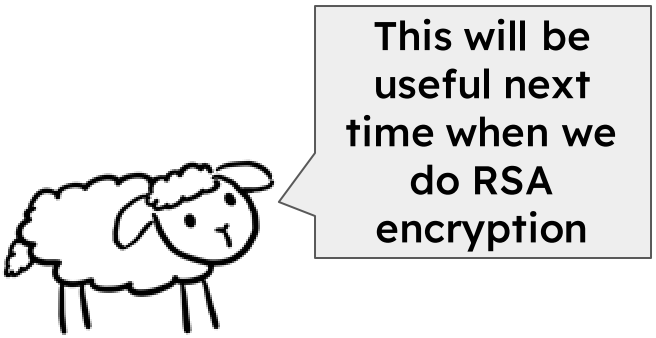
Input: Integers m, n > 0 with gcd(n, m)=1

Output: Mult. inverse of n (mod m) i.e. integer z so that $n \cdot z \equiv 1 \pmod{m}$

Algorithm:

1. Run **ExtendedEuclid**(n, m) to find (a, b) such that 1 = an+bm.
2. Return a.

Correctness: $1 \equiv an+bm \pmod{m}$, so $1 \equiv a \cdot n \pmod{m}$.



ExtendedEuclid (from HW 2):

Input: Integers $x \geq y \geq 0$, not both 0

Output: Triple (g,a,b) of integers where g = gcd(x,y) = ax+by.