# Lecture 3A: RSA

UC Berkeley EECS 70
Summer 2022
Tarang Srivastava

# Announcements!

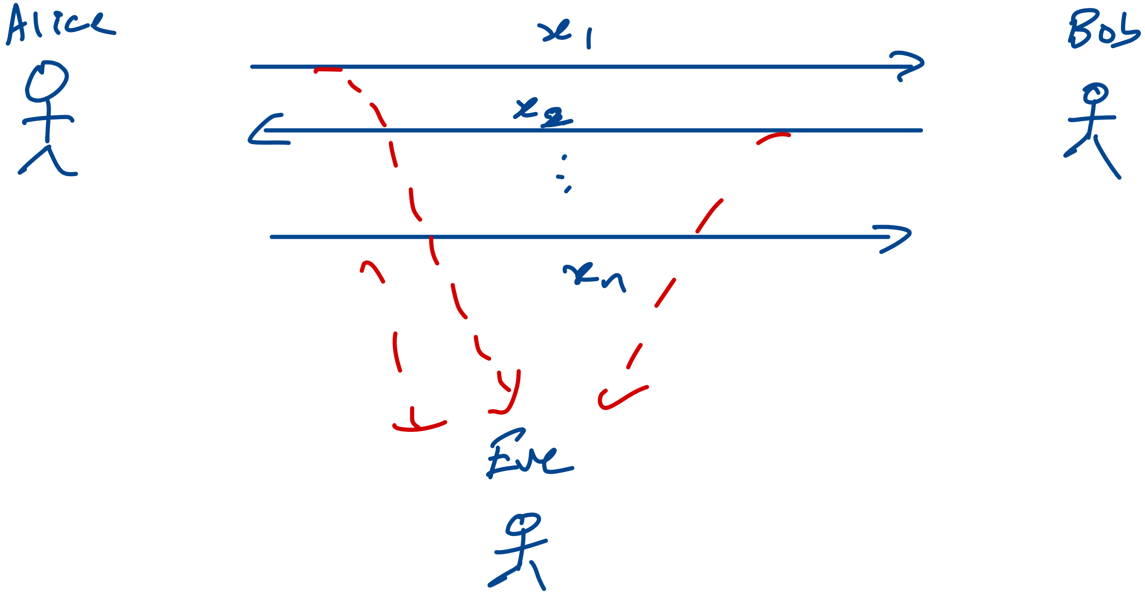
- Read the Weekly Post

→ HW1 Grades

- **HW 3** and **Vitamin 3** have been released, due **Thursday** (grace period Fri)
- HW 3 covers last Wednesday, Thursday and Today's lecture
- Any topic that's out of scope in this lecture will be in Orange.
  - You are not responsible for these topics, they're just here to give context
  - These topics will be covered in CS170 and CS161
- In this lecture, we will use small prime numbers as examples but in implementation we use large prime numbers (256 bits ≈ $10^{77}$ or more).

# Alice and Bob

Alice and Bob wish to send messages to each other **privately**.
Eve is able to intercept and read the messages.
How can Alice and Bob **encrypt** their messages, so even if Eve intercepts them she cannot understand them (i.e. **decrypt**).



"too few sequences"

"bruteforce"

# Using a Codebook

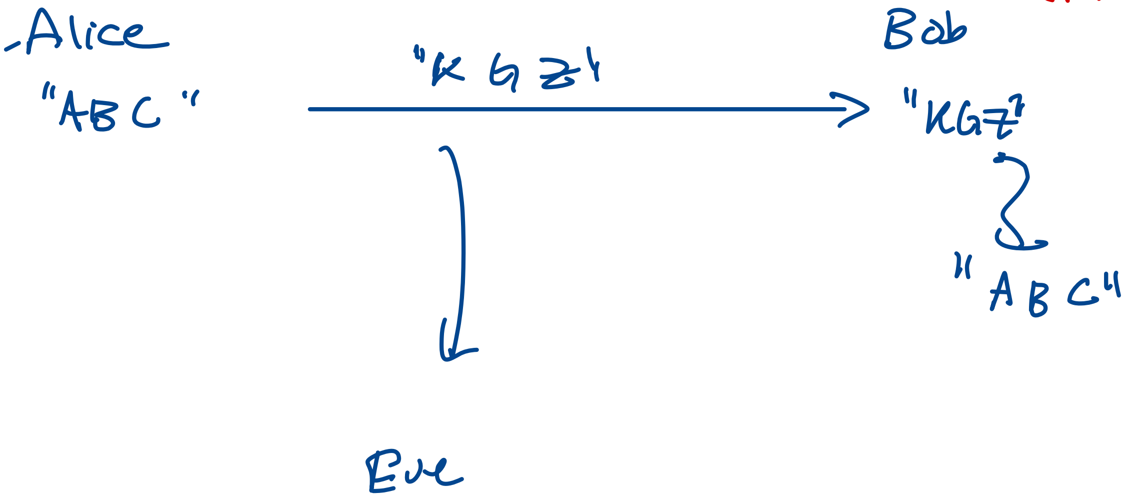
How can Alice and Bob **encrypt** their messages, so even if Eve intercepts them she cannot understand them (i.e. **decrypt**).

Problem: "Bob and Alice have to agree on a codebook before"

Carol → Oski

Codebook

A → K
B → G
C → Z
D → F
⋮
Z → B



# Public Key Cryptography

Alice generates

Not like a codebook!

Alice generates a **Public Key** (*K*), and a corresponding **Private Key** (*k*).
The public key *K* is known to everyone (including Eve), the private key *k* is known only to Alice.

Alice publishes this

Anyone can encode their message using the public key, and send it to Alice.
Only Alice knows the private key, so only she can decrypt the messages sent to her.

Everyone knows

$E_K$ := encryption function

$$\text{Bob} \xrightarrow{E_K(x) = y} \text{Alice}$$

message : $x$

Eve

$y \rightarrow$

$D_k$ := decrypting function

$D_k(y) = x$

How does Alice send a message to Bob?

Bob creates his own publickey

# RSA

Alice does the follow

## Setting up a Public Key

Pick two large primes $p$ and $q$. Let N = $pq$

Choose an $e$ that is coprime to the product $(p-1)(q-1)$ $\gcd(e, (p-1)(q-1)) = 1$

Compute the private key $d = e^{-1} \pmod{(p-1)(q-1)}$.

* Announce to the world the public key: $K = (N, e)$ → both to make a public key

Bob sends message

## Encrypting Messages

Let $x$ be your message. E(.) is the encryption function. Send $E(x) = x^e \pmod N$.

public key

Alice recieves a message

## Decrypting Messages

Let $y$ be the encrypted message. D(.) is the decryption function. $D(y) = y^d \pmod N$.

private key

## Why does this work?

Decrypting an encrypted message returns original message. $D(E(x)) = x = (x^e)^d \pmod N$

$D(E(x)) = D(x^e) = (x^e)^d = 1 \pmod N$

Can't ppl figure out p,q from N?

# Summary Questions

People

Alice, Bob, Eve

Alice has a public key (N,e)

Bob is sending a message x

Pick two large primes $p$ and $q$. Let N = $pq$
Choose an $e$ that is coprime to the product $(p-1)(q-1)$
Compute private key $k = d = e^{-1}$ (mod $(p-1)(q-1)$).
Announce to the world: K = (N, $e$)
Encryption: $E(x) = x^e$ (mod N).
Decryption: $D(y) = y^d$ (mod N).

RSA

| | N | $e$ | $p$ and $q$ | $d$ | Private Key | Public Key | Encryption Function | Decryption Function | $x$ message | $y$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Who knows | everyone | everyone | Alice | Alice | Alice | Everyone | Everyone | Alice | Bob (After Alice) | Everyone |
| Definition | $N = p \cdot q$ | randomly choose $e$ coprime to $(p-1)(q-1)$ | choose 2 large primes | $d = e^{-1}$ mod $((p-1)(q-1))$ | $K = (d, N)$ | $K = (N, e)$ | $E(x) = x^e$ (mod N) | $D(y) = y^d$ (mod N) | Bob just has a message | $y = E(x)$ |

# RSA Example

Pick two large primes $p$ and $q$. Let $N = pq$
Choose an $e$ that is coprime to the product $(p-1)(q-1)$
Compute private key $k = d = e^{-1} \pmod{(p-1)(q-1)}$.
Announce to the world: $K = (N, e)$
Encryption: $E(x) = x^e \pmod{N}$.
Decryption: $D(y) = y^d \pmod{N}$.

## Alice Setting Up Public Key

$p = 7$, $q = 11$ $\quad N = 7 \cdot 11 = 77$

$e$ coprime to $(7-1)(11-1) = 60$

$e = 7$ $\quad d = 43$

Public key $K = (N, e) = \boxed{(77, 7)}$

$7(0) + 60(1) = 60$
$7(1) + 60(0) = 7$
$7(-8) + 60(1) = 4$
$7(9) + 60(-1) = 3$
$7(-17) + 60(2) = 1$

$7^{-1} \equiv -17 \equiv 43 \pmod{60}$

## Bob Encrypting Message

$x = 2$ : message

$E(x) = E(2) = 2^7 \bmod 77$

$2^7 = 128 \equiv 51 \bmod 77$

$y = 51$

$$\xrightarrow{\quad 51 \quad}$$

## Alice Decrypting Message

$y = 51$ $\quad D(y) = D(51) = 51^{43} \bmod 77$

$\downarrow$ repeated squaring

$D(y) = 2 = x$ ☺

MAIN SLIDE

# Why does encryption/decryption work?

Thm: For every $x$ in {0, 1, ..., N-1}, $\mathbf{(x^e)^d \equiv x}$ **(mod N).** (i.e. D(E(x)) = x)

Proof:

Notice that $d \equiv e^{-1} \bmod (p-1)(q-1)$. So, $ed \equiv 1 \bmod (p-1)(q-1)$

$ed = k(p-1)(q-1) + 1$, $k \in \mathbb{Z}$.

$D(E(x)) = x$

We want to show that $x^{ed} \equiv x \bmod N \Rightarrow x^{ed} - x \equiv 0 \bmod N$

$x^{k(p-1)(q-1)+1} - x \equiv 0 \bmod N$. Since, p and q are prime and $N = p \cdot q$. If we show that $x^{k(p-1)(q-1)+1} - x \equiv 0 \pmod p$ and $\equiv 0 \pmod q$ then it is $\equiv 0 \pmod N$.

We wish to show $x^{k(p-1)(q-1)+1} - x \equiv 0 \pmod p$:

Case 1: $x$ is a multiple of $p$, then $p$ divides both the terms

Case 2: $x$ is not a multiple, so $x \in \{1, 2, \ldots, p-1\} \bmod p$

So by FLT $\underbrace{(x^{p-1})}_{\text{FLT}}{}^{k(q-1)} x - x \equiv 0 \pmod p$

$1^{k(q-1)} \cdot x - x \equiv x - x \equiv 0 \pmod p$

You can apply the identical argument to $q$. Thus, we are done.

FLT

$a \in \{1, 2, \ldots, p-1\}$

$a^{p-1} \equiv 1 \bmod p$

By CRT this answer of $x$ exists

# Why can't Eve reconstruct the Private Key?

Idea: $d = e^{-1}$ $(mod\ (p-1)(q-1))$, but Eve knows $e$ so why can't she just find the inverse?

Eve doesn't know $(p-1)$ or $(q-1)$ or even

$(p-1)\cdot(q-1)$

Eve only knows $N$

# Why can't Eve then figure out $p$ and $q$?

Eve knows N so why can't she figure out $p$ and $q$ using that?

We showed that every number has a prime factorization.
That is, given a natural number n there exist a unique set of primes such that $n$ is equal to their product.

NP HARD (CS170)

Finding this unique set of primes is **hard**.

What does it mean for a problem to be hard?
In this class, we will say that if the best solution is as good as guess-and-check it is hard.
To find the prime factorization you would have to try every factor for that number.

P vs. NP (CS170)

Is there a faster algorithm to find the factorization?
Unsolved Problem. It is possible with Quantum Computer.

Shor's Algorithm

# Why can't Eve just take the log?

Eve knows the public key (N, *e*). Eve then encrypts some message $y = x^e \ (mod\ N)$.
Then, the decryption is

Eve

$$x = y^d \ (mod\ N)$$

So, why can't Eve just do $log_y$ on both sides to **leak *d*** the private key?

This is called the discrete-log problem and it is **hard**. There is no known efficient solution for this problem.

Examples:

# How easy is it to find large primes?

**Theorem 7.3: [Prime Number Theorem]** Let $\pi(n)$ denote the number of primes that are less than or equal to $n$. Then for all $n \geq 17$, we have $\pi(n) \geq \frac{n}{\ln n}$. (And in fact, $\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1$.)

If we want a 512-bit prime number

Theorem says there is roughly 1 prime number every 355 numbers.

For 1024-bit numbers there's a prime every 710.

Just try random numbers and you will eventually find a prime number

You can efficiently check if a number is prime using the Miller-Rabin test. (CS170)

$O(\sqrt{n})$ CS61A

$\downarrow$

$O(\log n)$ CS170

# Can Even find a match using the encryption function?

If Bob encrypts some message $y = x^e$ ($mod$ N). Then, could Eve just plug in $x$' into the encryption function to find a match?

No! For 256-bit prime numbers that is $2^{256}$ it would take you 37 times the age of the universe to arrive at a guess for a $x' = x$.

# In Practice Sending Same Message Twice

Notice that since all the numbers are fixed, if you send the same message twice it will be encrypted the same way.

In practice, usually append a counter to the message so each message is unique.

$x + $ "1"

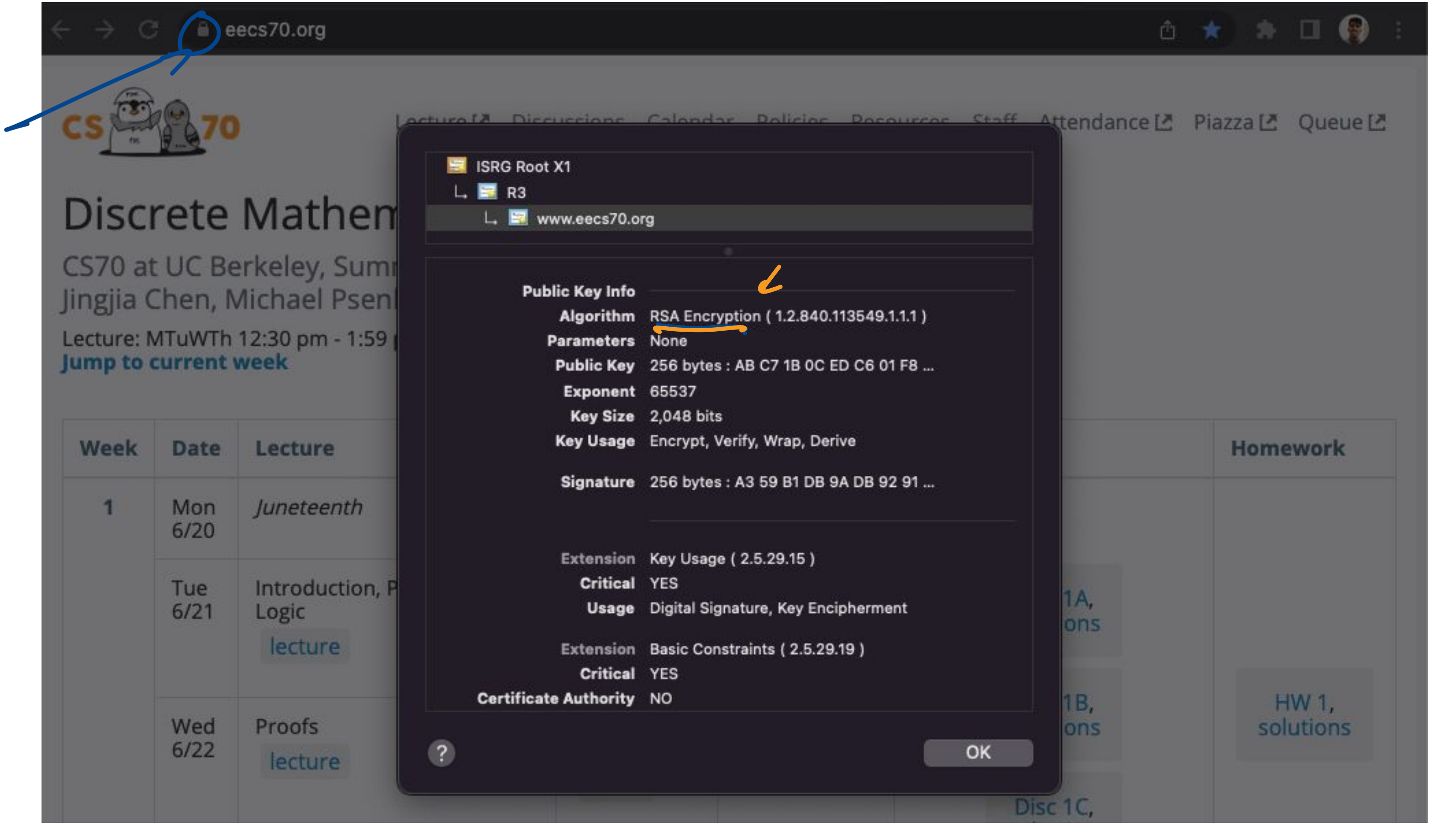
→

$x_2 + $ "2"

$x_3 + $ "3"

# You use some derivation of RSA every day

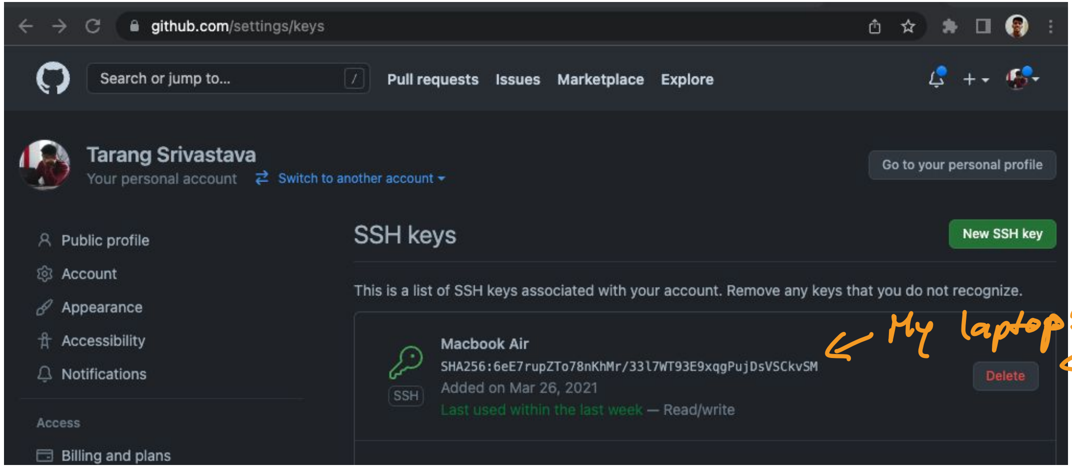
Click



ISRG Root X1
└ R3
└ www.eecs70.org

| | |
|---|---|
| **Public Key Info** | |
| **Algorithm** | RSA Encryption ( 1.2.840.113549.1.1.1 ) |
| **Parameters** | None |
| **Public Key** | 256 bytes : AB C7 1B 0C ED C6 01 F8 ... |
| **Exponent** | 65537 |
| **Key Size** | 2,048 bits |
| **Key Usage** | Encrypt, Verify, Wrap, Derive |
| **Signature** | 256 bytes : A3 59 B1 DB 9A DB 92 91 ... |
| Extension | Key Usage ( 2.5.29.15 ) |
| **Critical** | YES |
| **Usage** | Digital Signature, Key Encipherment |
| Extension | Basic Constraints ( 2.5.29.19 ) |
| **Critical** | YES |
| **Certificate Authority** | NO |

# You use some derivation of RSA every day



Tarang Srivastava

Your personal account ⇄ Switch to another account ▾

## SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

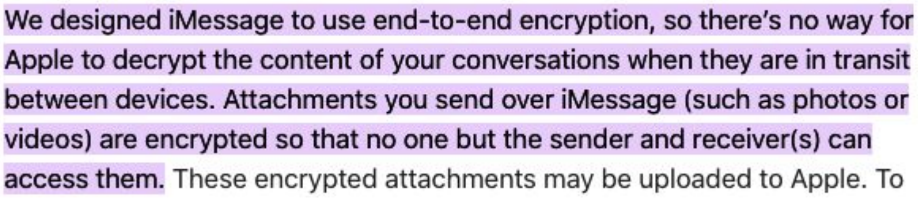
**Macbook Air**

`SHA256:6eE7rupZTo78nKhMr/33l7WT93E9xqgPujDsVSCkvSM`

Added on Mar 26, 2021

Last used within the last week — Read/write

← My laptop's public key



We designed iMessage to use end-to-end encryption, so there's no way for Apple to decrypt the content of your conversations when they are in transit between devices. Attachments you send over iMessage (such as photos or videos) are encrypted so that no one but the sender and receiver(s) can access them. These encrypted attachments may be uploaded to Apple. To

# A little story to end…

In 1977, Rivest, Shamir and Adleman publish the RSA algorithm you learned today.

Later that year, the British Intelligence Agency (GCHQ) declassify that they had developed the exact algorithm secretly in 1973.

Why do all this?

- Your company will ask you to make sure their data is secure
- You will want to make sure that your data is secure
- Most importantly, you have a **moral** responsibility to do so