

CS70 — SPRING 2026

LECTURE 9 : FEB. 17

Last Lecture

- Computing inverses : extended Euclid
- Chinese Remainder Theorem : solution to $\begin{cases} x \equiv a_1 \pmod{n_1} \\ \vdots \\ x \equiv a_k \pmod{n_k} \end{cases}$
- Fermat's Little Theorem : $a^{p-1} \equiv 1 \pmod{p} \quad \forall a \neq 0$
- Euler's Totient Theorem : $a^{\phi(n)} \equiv 1 \pmod{n}$
 $\forall a \text{ s.t. } \gcd(a, n) = 1$

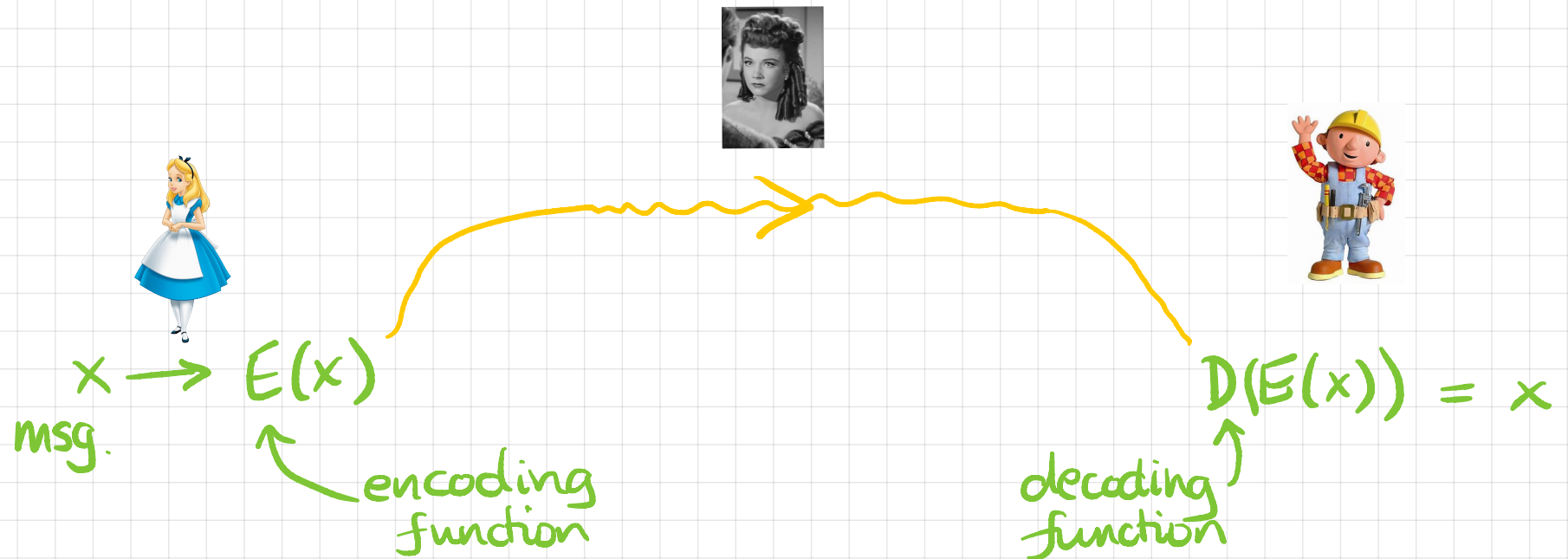
Today

- RSA Cryptosystem — an application of all the above

Cryptography

Alice wants to send a message to Bob
securely (nobody else can read it)

Eve is a malicious eavesdropper who
can monitor network traffic



Classical cryptography (pre-1970s)

Alice & Bob share in advance a secret (physical) "codebook" & use it for both encoding & decoding

Problems

- Eve might steal the codebook
- By observing a bunch of encrypted messages, Eve might break the code (cf Enigma in WWII)

Public-key Cryptography (Rivest-Shamir-Adelman, 1978)



publishes a public key K
keeps a related private key P



uses K to encrypt msgs to Bob



decrypts messages using P



learns nothing from observing encrypted msgs



"Padlock" analogy: anyone can lock it (with K)
need special key (P) to unlock





Chooses 2 large primes p, q [1024 bits $\approx 10^{308}$]

Chooses integer e coprime to $(p-1)(q-1)$

Publishes public key (N, e) where $N = pq$

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



To send a message x (an integer mod N) to Bob:

- computes $E(x) = x^e \pmod N$
 - sends $y = E(x)$ to Bob
-



Decrypts incoming message y via

$$D(y) = y^d \pmod N$$

Example: $p=5, q=17 \Rightarrow N=pq=85$
 $(p-1)(q-1)=64$

Choose $e=5$ (oprime to 64) $\Rightarrow$ Public key $K=(85, 5)$
 And $d=e^{-1} \pmod{64} = 13$ Private key

Message $x=11$ (+ve integer < 85)

$E(x) = 11^5 \pmod{85} = \boxed{61} = y$
 Send to Bob

$$\begin{aligned} 11^2 &= 121 \equiv 36 \\ 11^4 &\equiv 36^2 = 1296 \equiv 21 \\ 11^5 &= 11^4 \times 11 \equiv 21 \times 11 \\ &= 231 \equiv 61 \end{aligned}$$

$D(y) = 61^{13} \pmod{85} = \boxed{11}$

$$\begin{aligned} 61^2 &= 3721 \equiv 66 \\ 61^4 &= 66^2 = 4356 \equiv 21 \\ 61^8 &= 21^2 = 441 \equiv 16 \\ 61^{13} &= 61^8 \times 61^4 \times 61 \\ &\equiv 16 \times 21 \times 61 \\ &= 20,496 \equiv 11 \end{aligned}$$

Important: Alice never knows p, q or d !
 (or $(p-1)(q-1)$)

How do we know RSA works & is secure?

1. Correctness: Prove that $D(E(x)) = x \quad \forall x \in \{0, 1, \dots, N-1\}$
2. Security: Argue that Eve learns nothing by observing $x, E(x), N, e$
3. Efficiency: Show that all computations by Alice & Bob can be done efficiently
 - choosing large primes
 - computing $e, d, E(x), D(y)$

1. Correctness: Prove that $D(E(x)) = x \quad \forall x \in \{0, 1, \dots, N-1\}$

Proof:

Alternative Proof using Euler's Theorem

Recall: $x^{\varphi(m)} \equiv 1 \pmod{m} \quad \forall x \text{ s.t. } \gcd(m, x) = 1$
where $\varphi(m)$ is the number of such x

When $m = N = pq$, we have $\varphi(N) = pq - p - q + 1$
 $= (p-1)(q-1)$ WHY?

So by Euler's Thm:

$$x^{\text{ed}} = x^{k(p-1)(q-1)+1} \equiv x \pmod{N} \quad \forall x \text{ s.t. } \gcd(x, N) = 1$$

[And for the few special x with $\gcd(x, N) \neq 1$, we have $p|x$ or $q|x$ and can argue as before]

2. Security : Argue that Eve learns nothing by observing $x, E(x), N, e$



How could Eve try to learn x from $E(x) = x^e \bmod N$?

(i) Try guessing x

Note : $\exists$ unique x s.t. $E(x) = y$ so Eve can check

Problem : $2^{1024} \approx 10^{308}$ possibilities for x 😓

(ii) Try to figure out the factors p, q - then she could compute $d = e^{-1} \bmod (p-1)(q-1)$ and decrypt $x = y^d \bmod N$

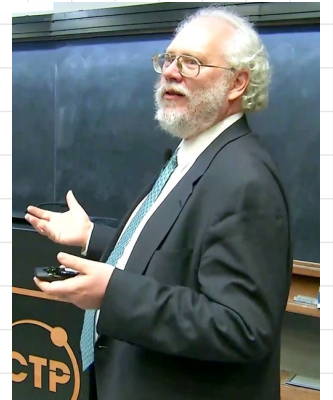
Problem : factoring large integers is very hard 😓

Note : computing $(p-1)(q-1)$ is equivalent to factoring N
since $(p-1)(q-1) = N - p - q + 1$

(iii) Try to solve the equation $x^e \equiv y \pmod{N}$ for x

Problem : Equivalent to the discrete log problem, which in turn is very hard 😓

Caveat : A quantum computer can (in theory) efficiently factorize large numbers (Shor's Algorithm)



→ can break RSA (and other similar public key cryptosystems)

⇒ New field of "post-quantum cryptography"

3. Efficiency: Show that all computations by Alice & Bob can be done efficiently

(i) computing e, d, E(x), D(y)

e: anything coprime to $(p-1)(q-1)$ - often $2^{16}+1$ (prime)

$d = e^{-1} \bmod (p-1)(q-1)$ - use extended Euclid

$$\left. \begin{aligned} E(x) &= x^e \bmod N \\ D(y) &= y^d \bmod N \end{aligned} \right\}$$

use repeated squaring: if e, d have ≤ 1024 bits then only need $\sim 2k$ multiplications

Multiplications mod N take $\sim (1024)^2 \sim 1m$ bit ops.

3. Efficiency: Show that all computations by Alice & Bob can be done efficiently

(ii) Choosing large primes (say, 1024 bits)

- pick a random 1024-bit number n
- test if n is prime
- if not, try again

Prime Number Theorem:

$$\frac{\text{\# of primes} < m}{m} \geq \frac{1}{\ln m}$$

"On average, 1 in every $\ln m$ numbers is prime"

Since $\ln 2^{1024} \approx 710$, at least 1 in 710 1024-bit nos. is prime!



$\Rightarrow$ expect ≈ 710 attempts to find a prime

Primality Testing : How do we test if a large number n is prime?

Recall : Factoring n is very hard

But : Testing if n is prime is (fairly) easy !

Bad Primality Test :

for $2 \leq a \leq \sqrt{n}$

if $a|n$ then output "n is not prime"

output "n is prime"

Running time : $O(\sqrt{n}) \approx 10^{154}$ if $n \approx 10^{308}$ 😓

Much Better: Fermat Test

pick $a \in \{2, 3, \dots, n-1\}$ at random

if $\gcd(a, n) \neq 1$ output "n is not prime"

if $a^{n-1} \not\equiv 1 \pmod{n}$ output "n is not prime"

else output "n is prime"

If n is prime, output is always correct

If n is not prime, test may output wrong answer...

This happens if the random a we pick is not a

witness : i.e., $\gcd(a, n) = 1$

and $a^{n-1} \equiv 1 \pmod{n}$

even though n is not prime

Good News : For almost all inputs n , at least half of the $a \in \{2, 3, \dots, n-1\}$ are witnesses !

This means the Fermat Test is correct with probability at least $1/2$

Try 100 different a 's $\Rightarrow$ Test is correct with prob.
 $\geq 1 - \frac{1}{2^{100}}$!

Carmichael Numbers : The inputs n that fail the Fermat Test are called Carmichael Numbers : they have no witnesses a .

First few CNs : 561, 1105, 1729, ...

Only ~ 1 in 2^{14} CNs up to 10^{21}

Miller-Rabin
Solovay-Strassen
⋮

Fact : There are efficient algorithms* (extensions of Fermat Test) that also work for CNs !

