Outline

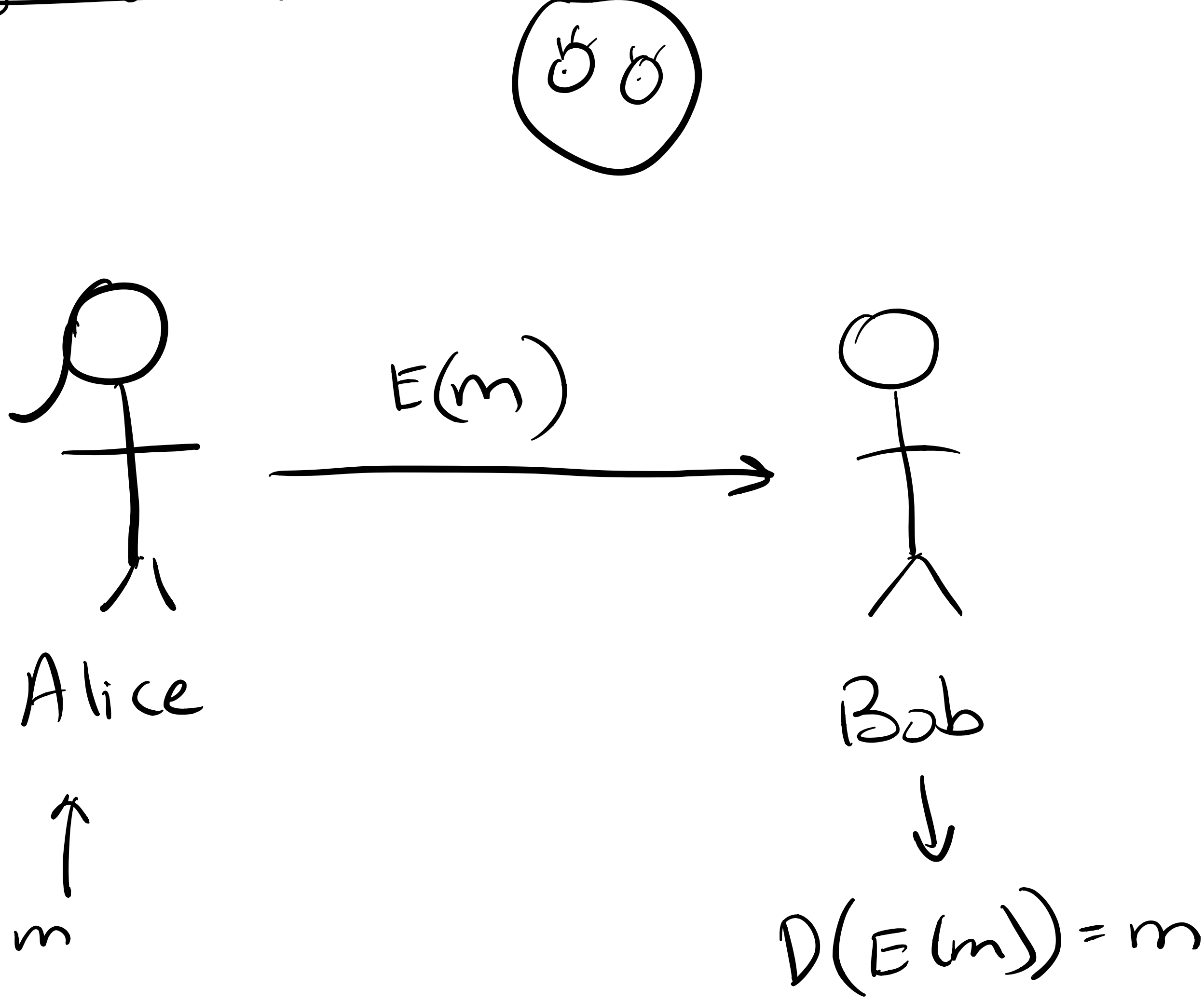
1. Cryptosystems
2. One-Time Pad
3. Public key Cryptography
4. FLT + RSA
5. Digital Signatures
6. Attacks

Reminder

Midterm on Monday July 12^{th} 8pm PDT

More announcements in the upcoming week.

1) Cryptosystems



- Alice wants to send a message (bitstring) to Bob

  She encrypts it and sends it as $E(m)$

- Eve can see $E(m)$

- Bob uses decryption function D to recover m

Note: E and D often depend on some key k

E_k and D_k

Goal: Make sure Eve cannot recover m, but BOB can.

(2) One-Time Pad

XOR : Exclusive OR, denoted $\oplus$

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

k is a bit string which is as long as m

Choose $E_k(m) = m \oplus k$

$D_k(m) = m \oplus k$

$$D_k(E_k(m)) = ((m \oplus k) \oplus k)$$
$$= m \oplus (k \oplus k)$$
$$= m$$

Only Alice and Bob can know k.

Pro:

It works if Eve does not know k

Cons:

Cannot reuse the pad k

Alice and Bob need to decide on k beforehand.

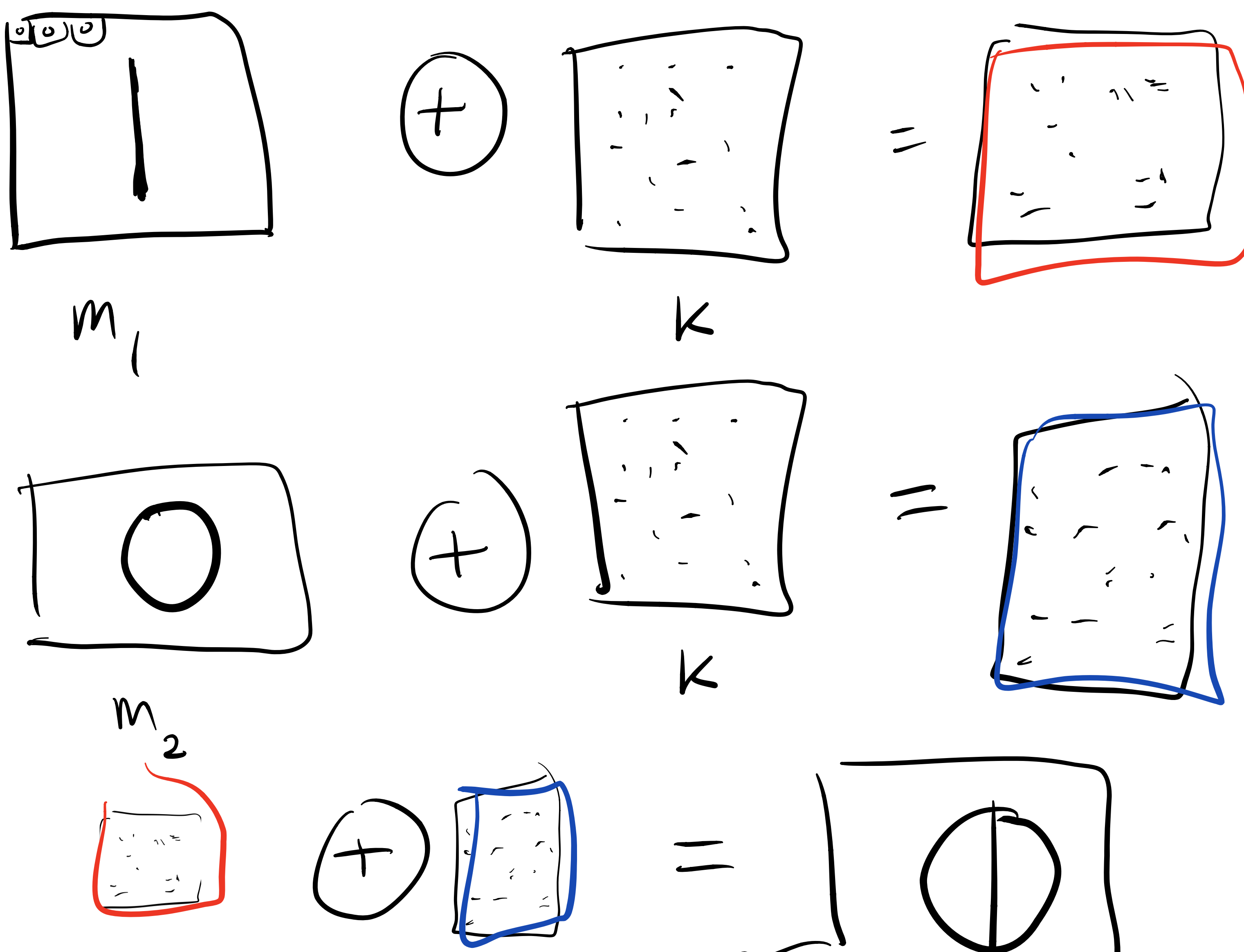
Why can't we reuse?

$$\left.\begin{aligned} E(m_1) &= m_1 \oplus k \\ E(m_2) &= m_2 \oplus k \end{aligned}\right\}$$ Eve can see these.

Eve can compute:

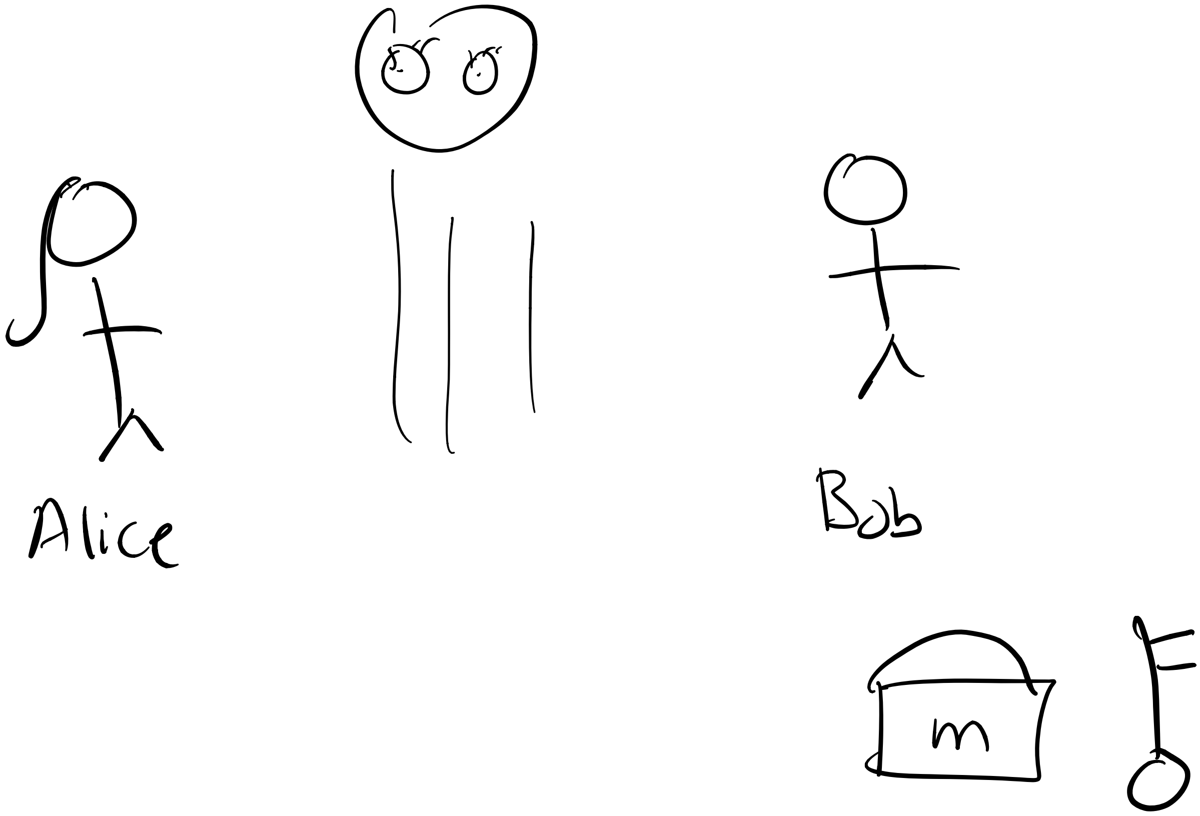
$$(m_1 \oplus k) \oplus (k \oplus m_2)$$

$$= m_1 \oplus m_2$$



Info leaked for m_1, m_2

③ Public key Cryptography



Can send messages securely without having to meet privately first!

④ <u>Fermat's Little Theorem</u> (FLT)

For any prime p and any $a \in \overbrace{\{1, \ldots, p-1\}}^{S}$

we have $a^{p-1} \equiv 1 \pmod p$

Proof

Observing that $f(x) = ax \pmod p$ is a bijection from S to S

This is because $\gcd(a, p) = 1$, so

$$ai \equiv aj \bmod p \xRightarrow{} i \equiv j \pmod p$$

multiply both sides by $a^{-1} \bmod p$

So, f maps each element of S to a distinct value in S

$$\Rightarrow \prod_{i \in S} i \equiv \prod_{i \in S} a \cdot i \pmod p$$

$$(p-1)! \equiv a^{p-1} (p-1)! \pmod p$$

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

RSA

Rivest, Shamir, Adleman

Start of with two primes p and q.

Public key (Treasure Box)

(N, e)

$N := pq$

e is a number such that $\gcd(e, (p-1)(q-1)) = 1$

Private key

$d := e^{-1} \pmod{(p-1)(q-1)}$

$E(x) = x^e \pmod{N}$

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

Correctness: $D(E(x)) = x$?

$$(x^e)^d \equiv x \pmod{N}$$

$$x^{ed} - x \equiv 0 \pmod{N}$$

Note: $ed \equiv 1 \pmod{(p-1)(q-1)} \Rightarrow ed = 1 + k(p-1)(q-1)$

$$x^{1+k(p-1)(q-1)} - x \equiv 0 \mod N$$

$$\underbrace{x\left(x^{k(p-1)(q-1)} - 1\right)} \equiv 0 \mod N$$

Approach: show divisiblity by p and by q separately.

Case 1: x is divisible by p

Case 2: $x\left(x^{k(p-1)(q-1)} - 1\right) \pmod p$

$$\equiv x\left(\left(x^{(p-1)}\right)^{k(q-1)} - 1\right) \pmod p$$

$$\Big\updownarrow \text{FLT}$$

$$\equiv x\left(1^{k(q-1)} - 1\right) \mod p$$

$$\equiv 0 \pmod p$$

Similarly, the expression is also divisible by q

So, it is divisble by $N = p \cdot q$

$$\Rightarrow x\left(x^{k(p-1)(q-1)} - 1\right) \equiv 0 \pmod N$$

Why does RSA work?

1. Assumes N is too large to brute force x^e for each x and check if the encoded message matches

2. Assumes d can't be computed without extracting p and q from N (factoring N is hard)

RSA Example (from Notes)

$p = 5$

$q = 11$

$N = 5 \cdot 11 = 55$

Say $e = 3$, $\gcd(e, 40) = 1$ ✓

Bob:

Public key: $(N, e) = (55, 3)$

Private key: $3^{-1} \bmod 40$

$$40 = 3 \cdot 13 + 1$$

$$40 \cdot 1 - 13 \cdot 3 = 1$$

$$d = -13 \equiv 27 \mod 40$$

Alice can then send x as $E(x) = x^3 \bmod 55$

Bob will decrypt this as $D(y) = y^{27} \pmod{55}$

Ex: $x = 13$ $\quad E(x) = 13^3 \pmod{55} = 52$

$$D(52) = 52^{27} \equiv 13 \mod 55$$

Digital Signatures

Trusted

You

Certificate Authority (CA)

Amazon.com

(N, e)

d

1. $m =$ "This is Amazon"
2. Signed by CA $= \underbrace{m^d}_{s}$
3. You can check using (N, e)

$$s^e \equiv m^{de} \equiv m \pmod{N}$$

Checks out, CA did confirm/sign.

RSA Attack

Replay Attack Example

I send $E(m)$ to Amazon to make purchase

Eve reads $E(m)$, and sends it to Amazon again.

Now I got charged twice :(

Solution

Send $E(\text{concatenate}(m, s))$ where

s is a random string.

If Amazon gets the same message twice,

it will just reject the second one.

RSA Sampling Primes

Prime Number Theorem states that

\# of primes $\leq N$ is at least $\frac{N}{\ln(N)}$

Go through all numbers less than N and check if they are prime.

There exists an efficient algorithm that tests if N is prime (polynomial time in the number of bits)

<u>Note:</u> Want p and q to be very large $\to$ 512 bits each.

$$x \equiv 20 \pmod{30}$$

$$x = 20 + 30k$$

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

$$\rightarrow \quad ed = 1 + k(p-1)(q-1)$$