HKN
CS 70 MT2 Review

Anjali Suresh
Gregory Junek
David Zhang

Topics covered

Counting

Probability Spaces, Conditional Probability

Countability, Computability

Mod math

RSA, Fermat’s Little Theorem

Polynomials, Secret Sharing

Error Correcting Codes

Counting

Count on Me

It’s late at night. You have 15 cartons of ice cream, each a different flavor. You decide you want to eat them all in one sitting.

How many orderings are possible?

Count on Me

It’s late at night. You have 15 cartons of ice cream, each a different flavor. You decide you want to eat them all in one sitting.

How many orderings are possible?

Ans: 15!

Count On Me

Explanation: With no decisions made, the initial ordering consists of 15 spaces.

The first carton can go in any one of these spaces, meaning there are 15 choices. This 15 must be multiplied by how many ways you can order the remainder.

Now, there are 14 remaining spaces for the second carton, and so on.

The final answer is 15 * 14 * 13 … * 1 = 15!

Count on Me

You’re not satisfied, so you decide to eat a real meal. There are 5 pieces of fried chicken, 4 sandwiches, 3 hot dogs, 2 breadsticks, and 1 jar of Nutella.

How many unique orderings are possible?

Count on Me

You’re not satisfied, so you decide to eat a real meal. There are 5 pieces of fried chicken, 4 sandwiches, 3 hot dogs, 2 breadsticks, and 1 jar of Nutella.

How many unique orderings are possible?

Ans: 15! / (1!*2!*3!*4!*5!)

Count On Me

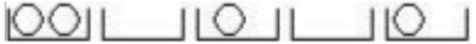
Explanation: Assuming all items of food are unique, we have the numerator, 15! possible sequences.

However, there are fewer than this because there are multiple identical copies of the same food - for example, a sequence (breadstick1, breadstick2 …) is indistinguishable from (breadstick2, breadstick1 …). Thus we must divide the numerator by the number of ways to rearrange the individual copies of each type of food in one of the numerator's sequences.
There are 5! ways to rearrange the fried chicken, 4! ways to rearrange the sandwiches, and so on. Following this pattern, we obtain the denominator.

Balls and Bins Recap

Consider k balls and n bins.

If you represent each ball as a 0 and bin divider as a 1, the number of ways to divide the balls among the bins is the same as the number of ways to order the balls and the bin dividers.

Ex:  as a bitstring is 00110110.

Balls and Bins Recap

In that case, 5 bins => 4 bin dividers.

Thus, n bins => n-1 bin dividers. The bitstring in question must have total length k + n - 1, with k 0s and n-1 1s.

We can choose (k + n - 1)C(k) locations for the 0s, and the locations for the 1s fall into place as a result, meaning that this is how many ways we can arrange k balls in n bins.

Buckets

You have 10 identical pieces of fried chicken and 4 empty buckets.

How many ways are there to allocate the fried chicken?

Buckets

You have 10 identical pieces of fried chicken and 4 empty buckets.

How many ways are there to allocate the fried chicken?

There are 4 “bins” and 10 “objects”. Represent each object as a 0, and each bin divider as a 1. There are 4 bins => 3 bin dividers.

The binary string is 13 bits long. Once we choose 3 bits to place the 1s in, we must necessarily place the 0s in the remaining bits.

There are 13C3 = 13C10 ways to do this.

Numbers (Fall ‘14 Final)

How many numbers are there from 0-999999 whose digits sum to 9?

Numbers (Fall ‘14 Final)

How many numbers are there from 0-999999 whose digits sum to 9?

There are 6 digits, which can be seen as 6 bins. The items in each of these bins must in total sum to 9. This translates into placing 9 identical objects (think of as 1s) into 6 bins, and in this way the total sum of the digits is forced to be 9.

6 bins => 5 bin dividers.
There are a total of 9 + 5 = 14 spaces - each can have either an object or a divider. Once we choose the 5 slots to put the dividers in, we have no choice where to put the remaining objects.

Ans: 14C5.

Numbers (Fall ‘14 Final)

How many numbers are there from 0-999999 whose digits sum to 19?

Numbers (Fall ‘14 Final)

How many numbers are there from 0-999999 whose digits sum to 19?

Like before, we have 19 objects and 6 bins. There are 19 + 5 = 24C5 ways to place these 19 objects in 6 bins.

But we’re working with base-10 numbers. Each digit can hold up to 9 in actuality, while in our example it can hold up to 19. We therefore need to subtract off the arrangements where a digit overflows (contains ten or more objects).

Numbers (Fall ‘14 Final)

How many numbers are there from 0-999999 whose digits sum to 19?

Only one digit can overflow at a time, because the smallest overflowing digit contains 10 objects, and if two digits contain ten objects then they sum to 20.

To overflow a digit, we pre-allocate 10 objects to that digit. There are 6 digits to choose from to overflow. After overflowing, we can do whatever we want with the remaining 9. We can divide this remaining 9 among all 6 bins, including the overflowing bin.

Numbers (Fall ‘14 Final)

Final Ans: There are 6*14C5 ways to overflow any digit of the number. These are invalid ways that we overcounted in our previous estimate.

Therefore, there are 24C5 - 6*14C5 numbers that sum to 19.

Probability!

Problem: Fruit Baskets (source: AIME)

A hotel packed breakfast for each of three guests. Each breakfast should have consisted of three types of rolls, one each of nut, cheese, and fruit rolls. The preparer wrapped each of the nine rolls and once wrapped, the rolls were indistinguishable from one another. She then randomly put three rolls in a bag for each of the guests. What is the probability that each guest gets one of each type of roll?

Solution: Fruit Baskets

Answer: $\dfrac{(9\cdot6\cdot3)(6\cdot4\cdot2)(3\cdot2\cdot1)}{9!} = 9/70$

To find the numerator, we count the number of ways in which each person can get one of each roll.

To find the denominator, we count the total number of ways in which the rolls can be arranged, which is 9!

Solution Cont. : Fruit Baskets

Consider the first person. Their first roll can be any one of the 9 available. Their second must be a roll of a different type, so one of 6, and their third must be one of the 3 remaining rolls of the last type. Thus there are (9*6*3) ways in which they can receive one of each type of roll.

For the second person, there are 6 total rolls (2 of each type of roll) available. Their first roll has 6 choices, the second has 4 and the third has 2.

For the last person, there will always be one of each, so there are 3! ways to arrange this.

Thus the numerator is (9*6*3) * (6*4*2) * (3*2*1).

Question: Casino Die

You want to play a game in which you bet whether the number on a rolled dice is 1, 2, or 3; or whether it is 4, 5, or 6. The casino you go to uses die that are 1.5 times as likely to land on an odd face than an even face. What is the probability of seeing a 1, 2, or 3?

Solution Part 1: The Model

We want the ratio of the probability of an odd number to an even number to be 1.5, or 3:2.

We thus say that each even number has probability 2p of being rolled and each odd number has probability 3p of being rolled.

Solution Part 2: The Algebra

We know that the sum of the probabilities of each roll must be 1, since there is a 100% chance that we will roll a number from 1 to 6.

$P(1) + P(2) + P(3) + P(4) + P(5) + P(6) = 1$

$3p + 2p + 3p + 2p + 3p + 2p = 1$

$15p = 1$

$p=1/15$

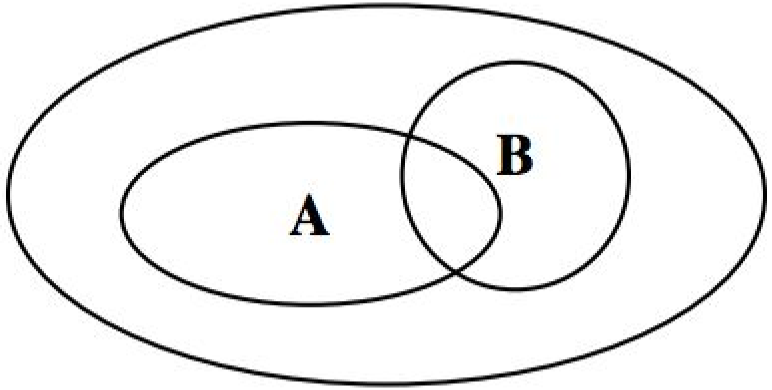
We want the probability of a 1, 2, or 3, so we have $3p+2p+3p = 8/15$

Recap: Conditional Probability

Pr(A|B) = the probability that A happens, given that B happens

= Pr(A,B) / Pr(B)

= the probability that both happen / the probability that B happens



Recap: Bayes' Rule

$\Pr(A \mid B) = \Pr(A, B) / \Pr(B)$

$\Pr(B \mid A) = \Pr(A, B) / \Pr(A) \rightarrow \Pr(A, B) = \Pr(A) * \Pr(B \mid A)$

$\Pr(A \mid B) = \Pr(A) * \Pr(B \mid A) / \Pr(B)$

Recap: Total Probability

$$\Pr(B) = \Pr(A, B) + \Pr(\sim A, B)$$

The probability of B happening is equal to the probability of A and B happening, plus the probability of not A and B happening.

$$= \Pr(B|A)*\Pr(A) + \Pr(B|\sim A)*\Pr(\sim A)$$

Question: Strange Coins

You decide you dislike the previous casino so you go to another and play a coin flipping game. The casino has two coins, one of which is fair and the other which has a probability of $\frac{2}{3}$ of being heads. You randomly pick a coin and flip it three times and see that it comes heads twice. What is the probability that the casino has played fairly and given you the fair coin?

Solution Part 1: The Model

Let the event that the casino gave you the fair coin be F, the event that the casino gave you an unfair coin be U, and let the event that you toss the coin three times and get 2 heads be S. You want to find $P(F|S)$.

Solution Part 2: The Algebra

Goal: P(F | S)

By Bayes' Rule: P(S | F)*P(F) / P(S)

Solution Part 2, Cont

To calculate the numerator:

P(S|F) = # of ways two coins can come up heads * P(2 heads, 1 tail)

= 3C2 * (½)^3

P(F) = ½, since you randomly pick either the fair coin or the unfair coin

3C2 * (½)^3 * ½ = 3/16

Solution Part 2, Cont

To calculate the denominator, P(S):

Recall P(S) = the probability that you get two heads out of three tosses.

By the Total Probability Rule, $P(S) = P(F, S) + P(U, S) = P(S|F)*P(F) + P(S|U)*P(U)$

Solution Part 2, Cont

$P(S|U) = 3C2 * (\frac{2}{3})^\wedge 2 * (\frac{1}{3})$

$P(U) = \frac{1}{2}$

$P(S|U) * P(U) = 3 * 4/9 * \frac{1}{3} * \frac{1}{2} = 2/9$

Solution, Part 2, Cont

Final Ans:

$$\frac{\binom{3}{2}\cdot\left(\frac{1}{2}\right)^3\cdot\frac{1}{2}}{\binom{3}{2}\cdot\left(\frac{1}{2}\right)^3\cdot\frac{1}{2}+\binom{3}{2}\cdot\left(\frac{2}{3}\right)^2\cdot\left(\frac{1}{3}\right)\cdot\frac{1}{2}}$$

3/16 / (3/16 + 2/9)

Countability

Color Countability

Are the following sets finite, countably infinite, or uncountable?

1. The set of all colors representable in RGB (each primary color has an integer value between 0 and 255).
2. The set of all colors.
3. A function Colorify = $\{f : \mathrm{N} \to \text{color}\}$ maps each natural number to a distinct color. The set of all colors that are output by Colorify.
4. All 116000 people in Berkeley have a favorite color. The set of all functions $\{f : \text{person in Berkeley} \to \text{their favorite color}\}$.

Color Countability: Solution

Are the following sets finite, countably infinite, or uncountable?

1. The set of all colors representable in RGB (each primary color has an integer value between 0 and 255). Finite
2. The set of all colors. Uncountably infinite
3. A function Colorify = $\{f: \mathbb{N} \to \text{color}\}$ maps each natural number to a distinct color. The set of all colors that are output by Colorify. Countably infinite
4. All 116000 people in Berkeley have a favorite color. The set of all functions $\{f: \text{person in Berkeley} \to \text{their favorite color}\}$. Uncountably infinite

Pi Countability

Is the following set finite, countably infinite, or uncountable?

5a. The number of *k*-length substrings for some natural number *k* of pi = 3.14159265358…

5b. The number of finite-length substrings of pi = 3.14159265358…

Pi Countability: solution

5a. The number of *k*-length substrings for some natural number *k* of pi = 3.14159265358… Finite (10^k)

5b. The number of finite-length substrings of pi = 3.14159265358… Countably infinite

Every substring of pi has a natural number start index, and a natural number end index (or length). Thus, there are ($N \times N$) substrings, countably infinite.

(This question assumes pi is normal -- informally, that all digits and sets of digits occur with the same frequency).

Computability

Uncomputable numbers

Show that there exist infinitely many uncomputable numbers: numbers that cannot be output by a computer program.

Uncomputable numbers: solution

Every computer program, as well as every input can be represented as a finite-length bit string {0, 1}*. So there are countable program-input pairs. So there are countable outputs to computer programs.

There are uncountably many real numbers, so even when removing the outputs of all computer programs, uncountably many real numbers remain that are not the output of any computer program.

Number-finding programs (from sp14 hw)

a. Can you write a program that gets *n* (a natural number) as input and finds the shortest formula that computes *n*? A formula is a valid sequence consisting of decimal digits, the operators +, ×, ˆ (raising to the power), and parentheses.
b. Can you write a program that gets *n* and finds the shortest computer program that computes *n*?
c. Can you write a program that gets *n* and finds a computer program *Q* that computes *n*? *Q* must have the following property: of all the programs that compute *n*, it must have the shortest (length + execution time).

Number-finding programs: solution

a. Can you write a program that gets *n* (a natural number) as input and finds the shortest formula that computes *n*? A formula is a valid sequence consisting of decimal digits, the operators +, ×, ˆ (raising to the power), and parentheses.

Yes: first, note that there is definitely a formula of length at most $\log_{10} n + 1$ -- the number itself (for example, the formula for n = 1000 would be “1000”, which has a length of 4). Thus, without fear of looping infinitely, compute all *i*-length formulas, in order of increasing *i*, tossing out all syntax errors. Return the first formula that evaluates to *n*.

Number-finding programs: solution

b. Can you write a program that gets *n* and finds the shortest computer program that computes *n*?

No: there is no way to know for sure what a program's output will be. Say we've found a program with a length of 50 characters which computes *n*. Perhaps some length-49 program would compute *n*, but it seems to us that it's looping. Our program can't be sure whether it infinitely loops, or whether it will eventually return *n*.

Number-finding programs: solution

c. **Yes:** first, note that there is a program, "return n", that accomplishes this in some time + length value *V*. Now, iterate through all programs of length less than V. Throw out the programs that do not compile. Now, for all remaining programs, of length *L*, run them on a timer, allotting (V - *L*) time to run. If any return *n*, return the one of these with the shortest *L* + *t* value; otherwise, return the default "return n."

Modular Arithmetic

Modular Arithmetic

Prove that if:

$x \cong 1 \bmod 6$

then:

$x \cong 1 \bmod 2$

$x \cong 1 \bmod 3$

Modular Arithmetic

Prove that if:

$x \cong 1 \bmod 6$

then:

$x \cong 1 \bmod 2$

$x \cong 1 \bmod 3$

By definition of equivalence, we know that there exists an integer “a” such that:
$x = 6a + 1$.

After factoring, we get:
$x = 2(3a) + 1$.

a is an integer, so 3a is an integer. Call it “b”. We have:
$x = 2b + 1$.

Therefore, $x \cong 1 \bmod 2$.

Similarly, we have:
$x = 6a + 1 = 3(2a) + 1 = 3c + 1$.

Therefore, $x \cong 1 \bmod 3$.

Multiplicative Inverse (EGCD)

Compute the multiplicative inverse of 5 mod 18.

Multiplicative Inverse (EGCD) - Solution

Compute the multiplicative inverse of 5 mod 18.

IMPORTANT: Check if multiplicative inverse exists.
Condition: gcd(18, 5) == 1.

Multiplicative Inverse (EGCD) - Solution

Compute the multiplicative inverse of 5 mod 18.

Right Columns: Start from (0,1) at the bottom. For given (a,b), solve for (x,y) such that ax + by = 1

Left Columns: GCD(18,5)

a	b	x	y
18	5	2	-7
5	3	-1	2
3	2	1	-1
2	1	0	1
1	0	1	0

Answer: $5^{-1} = -7 \cong 11 \bmod 18$

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 3$

$x \cong 2 \bmod 4$

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 3$

$x \cong 2 \bmod 4$

We first need to check that the moduli are coprime.

The moduli of the first and third equations are not coprime! We need to check whether they are consistent with each other.

If they are not consistent, there is no solution.

If they are consistent, we remove the least constraining equations.

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 3$

$x \cong 2 \bmod 4$

Let “a,b” be arbitrary integers. We use the definition of modularity to rewrite the constraints as:

$x \cong 1 \bmod 2 \rightarrow x = 2a + 1$

$x \cong 2 \bmod 4 \rightarrow x = 4a + 2$

We factor the second constraint.

$x = 4a + 2 \rightarrow x = 2(2a + 1) \rightarrow x = 2(2a + 1) + 0$.

Let $c = 2a + 1$. a is an integer, so c is also an integer.

$x = 2c + 0 \rightarrow x \cong 0 \bmod 2$. Inconsistent with our original first constraint, so there is no solution!

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 5$

$x \cong 1 \bmod 6$

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 5$

$x \cong 1 \bmod 6$

We first need to check that the moduli are coprime.

The moduli of the first and third equations are not coprime!
We need to check whether they are consistent with each other.

If they are not consistent, there is no solution.

If they are consistent, we remove the least constraining equations.

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 5$

$x \cong 1 \bmod 6$

We decompose the third equation into its prime factors.

$x \cong 1 \bmod 6 \rightarrow x \cong 1 \bmod 2,\ x \cong 1 \bmod 3.$

We now have 4 constraints:
$x \cong 1 \bmod 2$
$x \cong 2 \bmod 5$
$x \cong 1 \bmod 6 \rightarrow x \cong 1 \bmod 2,\ x \cong 1 \bmod 3.$

The first and third constraints are not coprime, but they are consistent with each other, so we can remove one.

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 5$

$x \cong 1 \bmod 6$

$x \cong 1 \bmod 2$
$x \cong 2 \bmod 5$
$x \cong 1 \bmod 3.$

The moduli are now coprime. Use CRT to find x using the new constraints.

Chinese Remainder Theorem

Find a value of “x” that satisfies the following constraints:

$x \cong 1 \bmod 2$

$x \cong 2 \bmod 5$

$x \cong 1 \bmod \cancel{6}\ 3$

$x = 1 * (5*3) * ((5*3)^{-1} \bmod 2) + 2 * (2*3) * ((2*3)^{-1} \bmod 5) + 1 * (2*5) * ((2*5)^{-1} \bmod 3)$

Remember: This answer is valid modulo the least common multiple of the constraint moduli.

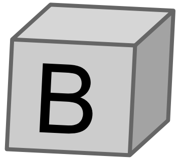
LCM(2,5,3) = 30.

Answer: $x \cong 7 \bmod 30$.

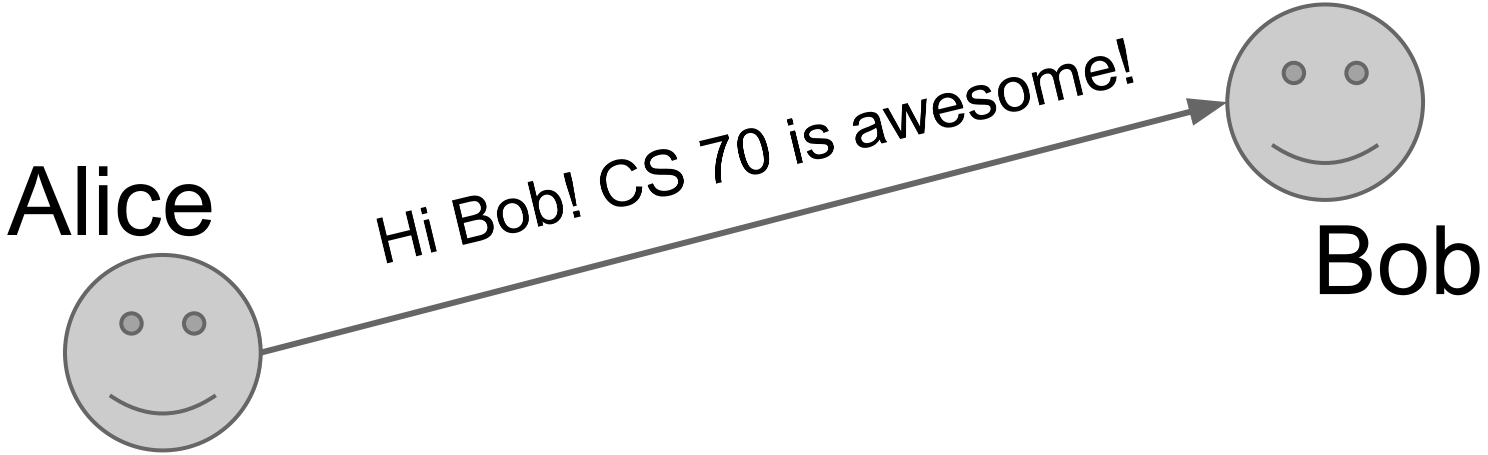
RSA

RSA — Parameters

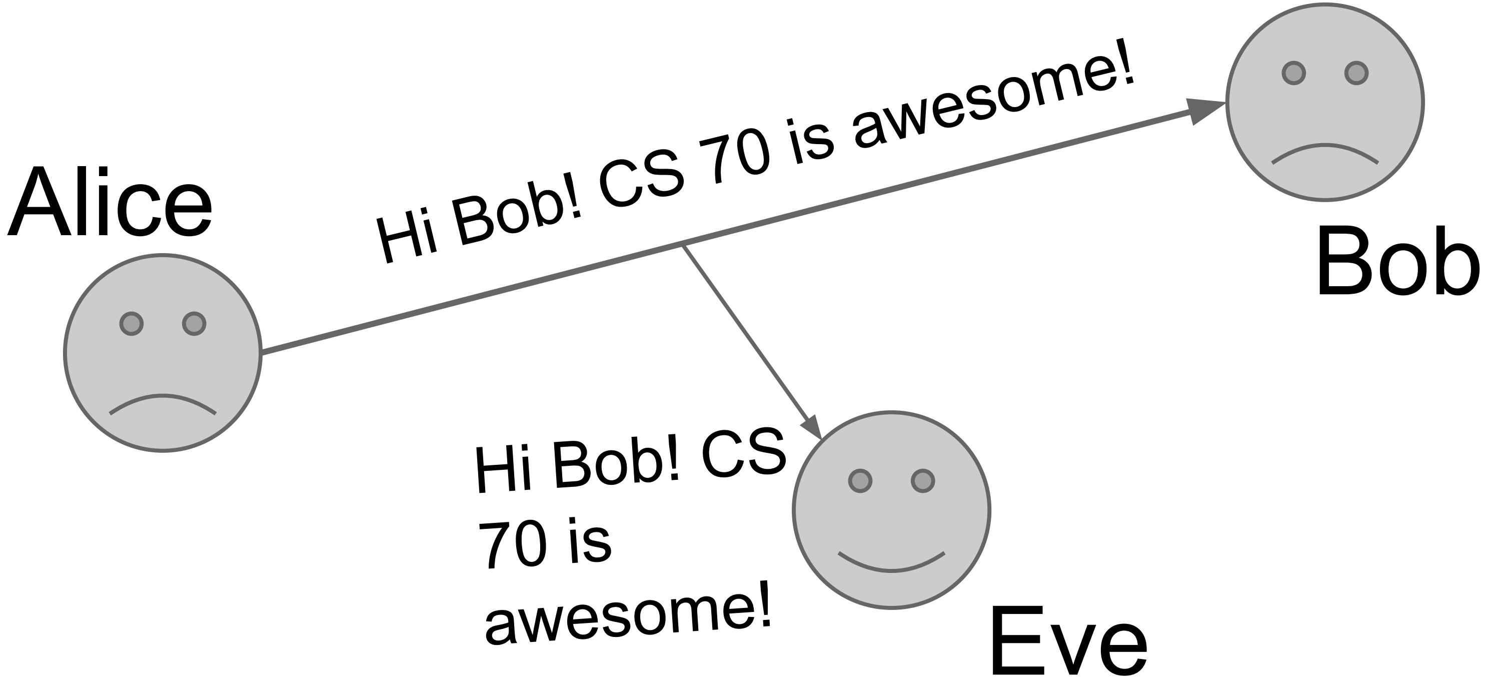
p	large prime	
q	large prime	
$N = pq$	(messages, x, are sent mod N)	
e	small, relatively prime to $(p\text{-}1)(q\text{-}1)$	
$d = e^{-1} \bmod (p\text{-}1)(q\text{-}1)$		Private Key
(N,e)		Public Key



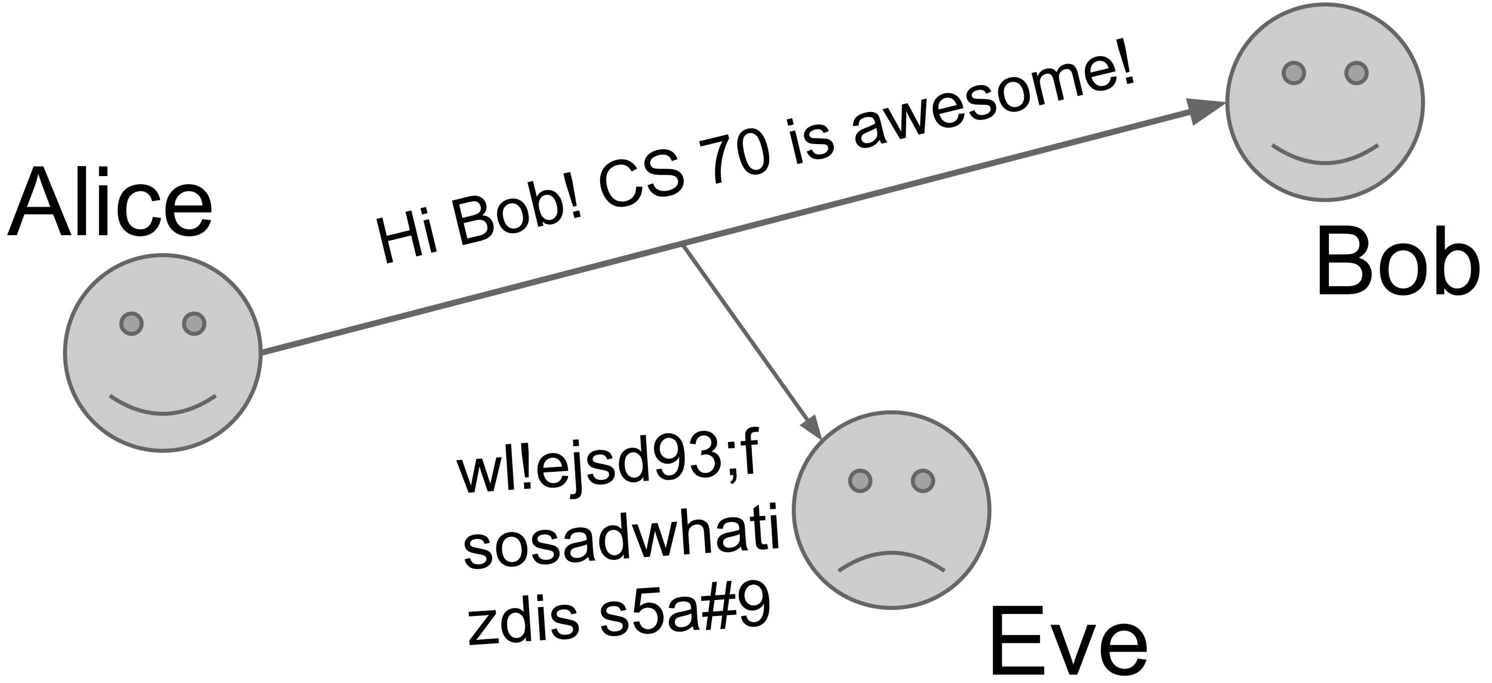
Setup



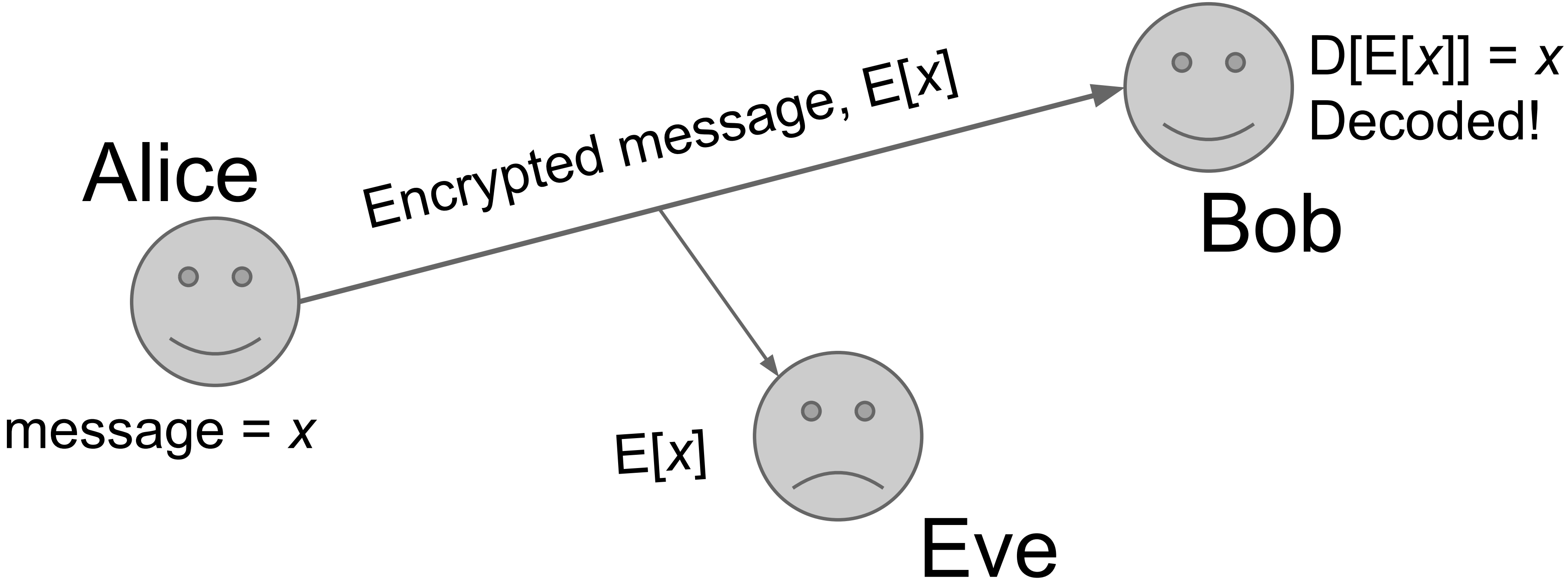
Setup



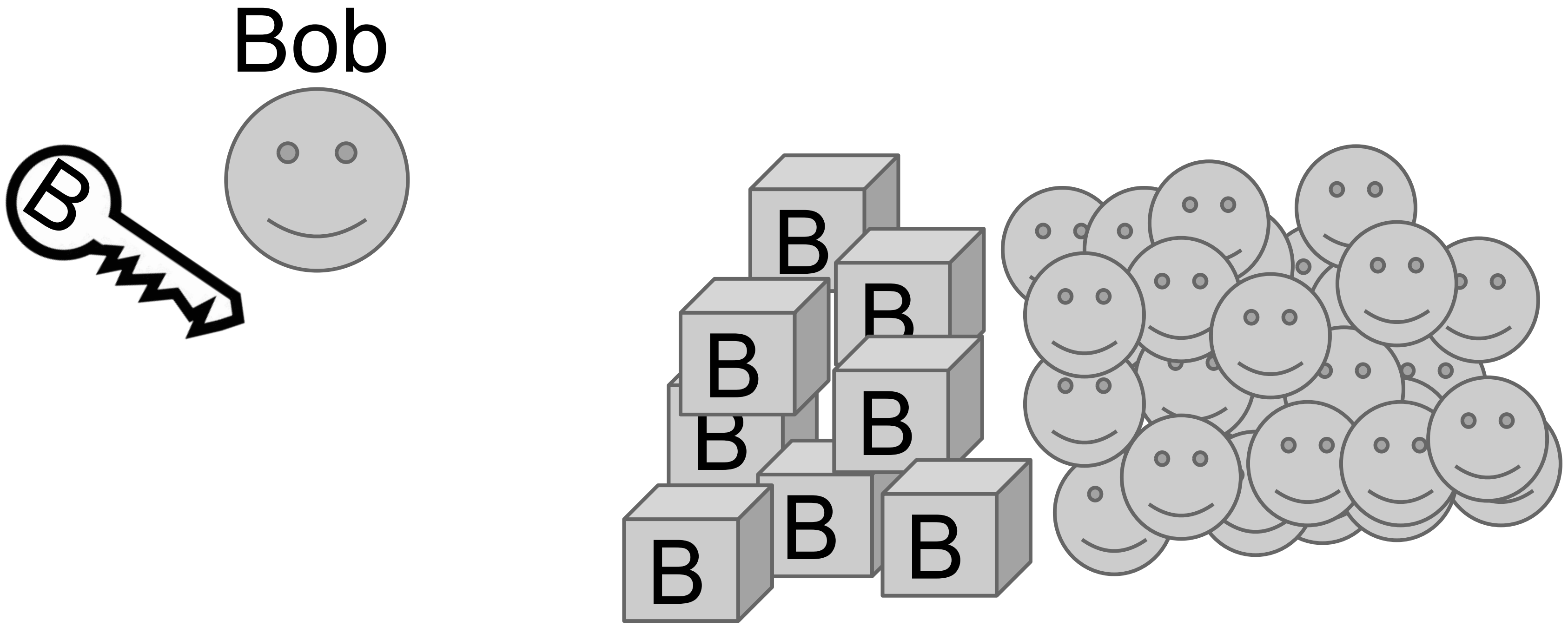
We Want


Alice
Hi Bob! CS 70 is awesome!
Bob
wl!ejsd93;f
sosadwhati
zdis s5a#9
Eve

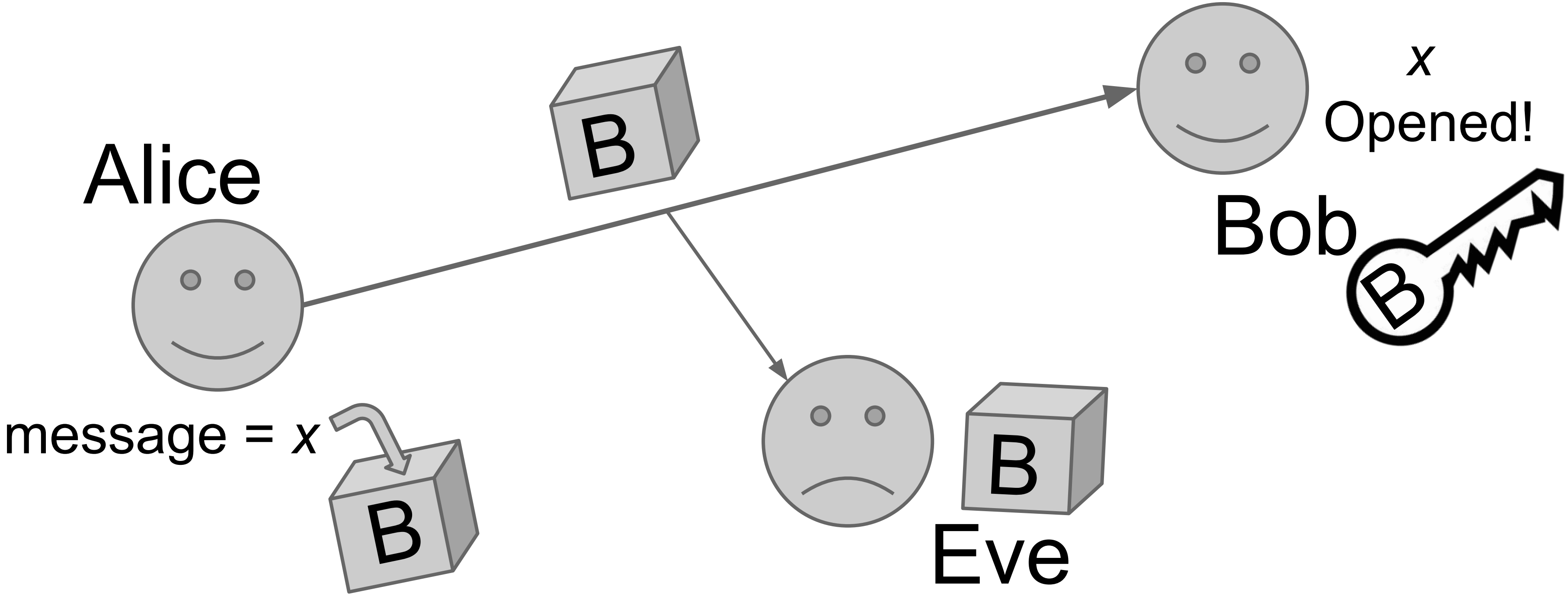
We Want


Alice
message = x
Encrypted message, E[x]
Bob
D[E[x]] = x
Decoded!
E[x]
Eve

Public Key Crypto



Public Key Crypto

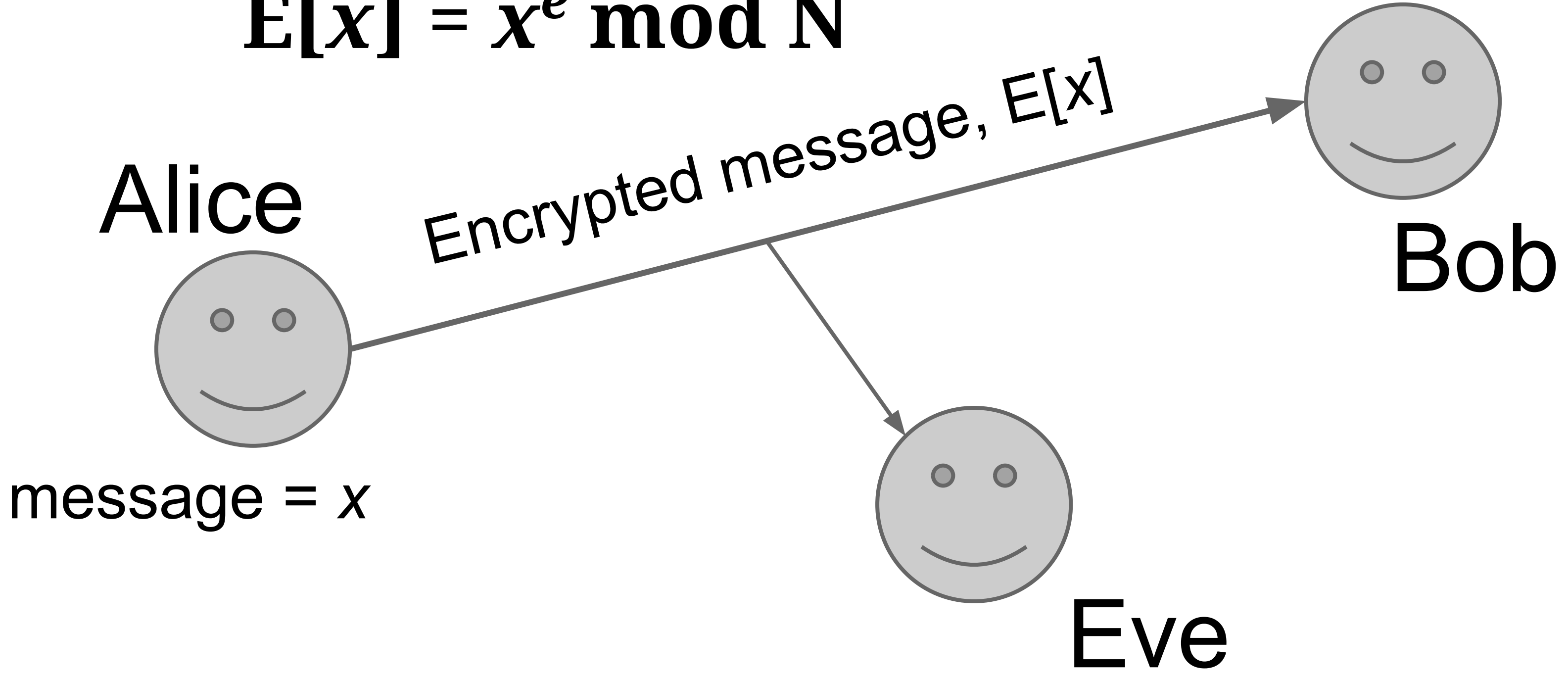


Check!

Will d always exist? Why?

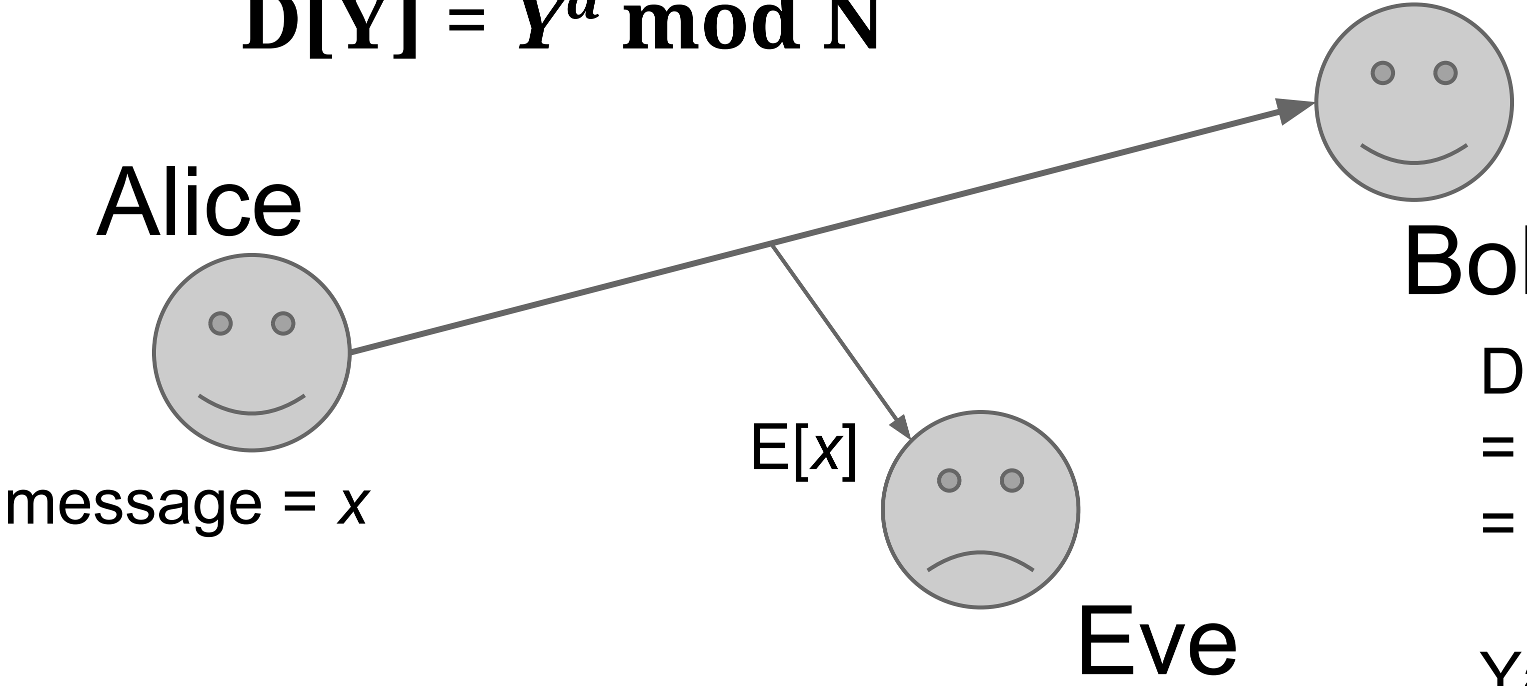
RSA—Encryption

$$\mathbf{E}[x] = x^e \ \mathbf{mod\ N}$$



RSA—Decryption

$$\mathbf{D[Y]} = \boldsymbol{Y^d} \ \mathbf{mod\ N}$$



$D[E[x]]$
$= (x^e)^d \bmod N$
$= x$

Yay!

Warm-up

Let $p = 13$, $q = 17$, $e = 5$

What should d be?

If $x = 3$, what should you transmit?

Answer

d = 77

Transmit x^5 = 22 mod 221

Magic Command (adapted from sp14 MT2)

(See your worksheet for context).

If “0011011010” is typed into a keyboard as input to a machine, the Earth will blow up. Everyone knows this number.

You’re asked to add some security to the machine. Design a module that transforms the input from the keyboard and transmits it to the machine. Anyone can look into the module and see exactly what it does. Nonetheless, the module should work so that authorized personnel can still blow up the Earth (if the need arises) but no one else can.

Magic Command: solution

Use RSA, with a secure p and q, and secret d.

The module raises the input to the power (e mod N), both known to everyone, and transmits the result to the machine.

If authorized personnel want to blow up Earth, they transmit (0011011010 ^ d mod N). No one else can compute this number.

Redundant RSA

Joe Hacker decides that he wants to have two public-private key pairs to be used with RSA -- that way if one of his private keys is compromised, half his communication is still secure.

For convenience, he decides to use a common composite n = pq (p,q primes) and selects two separate encryption exponents e1 and e2, giving two distinct decryption keys d1 and d2. He makes e1, e2, and n public. You can assume that e1 and e2 are relatively prime.

Suppose two people send the same secret plaintext message m to Joe, one encoded by e1 and the other with e2. Explain how an evil eavesdropper can efficiently determine m if he intercepts both these messages. What is the running time of your algorithm? (Hint: use the fact that e1 and e2 are relatively prime).

Redundant RSA Solution

We have access to x = m^{e1} mod N and y = m^{e2} mod N, the two encrypted texts. But what we really want is m^1 mod N. How do we get a 1 out of e1 and e2?

Because they are relatively prime, EGCD will give us integers a and b such that a*e1 + b*e2 = 1. Once we find a and b, we compute

$x^a * y^b = m^{(a * e1)} * m^{(b * e2)}$ mod N = $m^{(a * e1 + b * e2)}$ mod N

= m mod N = m.

Polynomials

Basics

- ➔ $p(x) = a_d x^d + a_{d-1} x^{d-1} \ldots + a_0$.
- ➔ If $a_d \neq 0$, d is the degree of p(x).
- ➔ c is a zero of polynomial p if p(c)=0.

Properties

- ➔ **If p is not a zero polynomial and is of degree d, it has d roots. (It can be repeated roots.)**

- ➔ **A degree d polynomial is uniquely defined by d+1 points. Think of (x, p(x)).**

Interpolation

➔ Say we have (d+1) points of a polynomial p.

➔ How do we construct a polynomial that fits these points?

➔ Let the points and evaluations be $(a_i, p(a_i))$, where i can go from 0 to d.

- ➔ Consider the following construction for i = 0.
- ➔ $D_0(x)=((x - a_1)(x - a_2) \dots (x - a_d))/ ((a_0 - a_1)(a_0 - a_2) \dots (a_0 - a_d))$
- ➔ Now, note that $D_0(a_i)$ for any $i \neq 0$ is zero, and when i=0, we get that $D_0(a_0) = 1$.
- ➔ We can do this for all i!

- ➔ $p(x) = p(a_0)D_0(x) + p(a_1)D_1(x) + p(a_2)D_2(x) \dots + p(a_d)D_d(x)$
- ➔ We have managed to construct a polynomial that pases through all the points!
- ➔ But is it unique?
- ➔ Yeah! Think about how you can prove it!

Polynomial Division

- ➔ If q divides p, q must have a degree that is less than or equal to p!
- ➔ The remainder r will always have degree smaller than p!
- ➔ You can use this and induction to prove property 1!

Polynomial Applications!

- ➔ Polynomials are cool so we do cool stuff with it.
- ➔ Secret Sharing!
- ➔ Error Correcting Codes!

Secret Sharing

- ➔ We know that we need (d+1) points to recover a polynomial of degree d.
- ➔ Say we have a secret that we need at least k people to be able recover.
- ➔ Polynomials to the rescue!

- ➔ Define polynomial p of degree (k-1)
- ➔ Let p(0) = secret.
- ➔ Give (1, p(1)) to person 1, (2, p(2)) to person 2 and so on!
- ➔ When they meet, they'll be able to interpolate and recover p!
- ➔ They then just need to evaluate p(0)

- ➔ You can also build hierarchies!
- ➔ Let A and B be groups of people where people in A are more important than people in B.
- ➔ Let p be the secret split among people of A.
- ➔ Let one evaluation of p be the secret to be shared among B!

Error Correcting Codes

- ➔ There are erasure errors and there are general errors!
- ➔ Erasure errors are easier to handle.

Erasure Errors

- ➔ Polynomials!
- ➔ Let our message be n points long. Let a_i be the value of our message at position i.
- ➔ We can define a polynomial p of degree n-1 that passes through all the above points!

- ➔ We then need to send n+k evaluations of that polynomial.
- ➔ When k get dropped, the remaining n points can be used to interpolate back p!

General errors!

- What if someone changed our message? Specifically, what if k points have been changed?
- What do we do then?
- Create a polynomial p of degree (n-1) that passes through your message, and send n+2k points!

- ➔ Let the received message in position i be r_i.
- ➔ Consider the following polynomial definition: $E(x) = (x - e_1)(x - e_2) \dots (x - e_k)$ where e_i is the location of error i.
- ➔ Note that E(x) has degree k.
- ➔ Let $Q(x) = P(x)E(x)$. Q now has a degree of $(n-1) + k = n + k - 1$.

- ➔ Now, since we don’t know P, we can let $Q(x) = a_{n+k-1}x^{n+k-1} + \ldots + a_0$.
- ➔ Now consider position 1.
- ➔ Q(1) gives us the following equation.
- ➔ $a_{n+k-1} + \ldots + a_0 = r_1E(1) = r_1(1 - e_1)(1 - e_2) \ldots (1 - e_k)$
- ➔ We can make n+2k such equations since our message is n+2k points long!

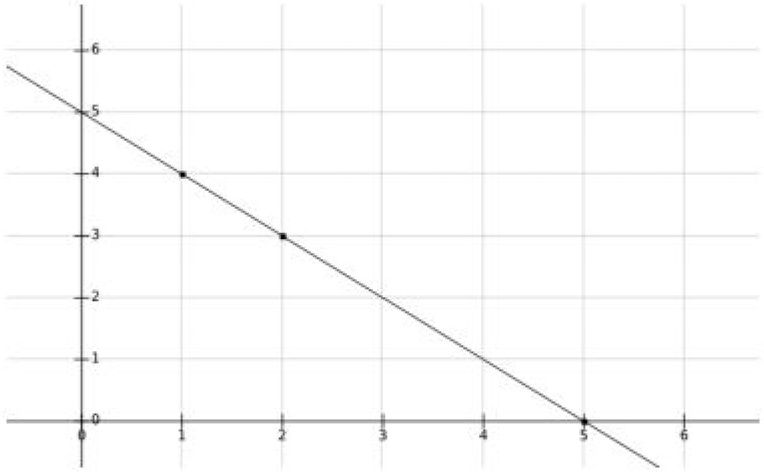
- ➔ Q has n+k unknowns. E has k unknowns. We have a total of n+2k unknowns.
- ➔ We also have n+2k equations.
- ➔ Linear algebra!
- ➔ Once we solve the equations, P = Q/E.

Question [MT2 Fa14]

Find the lowest degree polynomial that passes through (1, 4), (2, 3) and (5, 0) in mod 7.

Answer

6x + 5! Since we are working in a finite field, it is possible to simply graph it out.



Question [MT2 Fa14]

You would like to send a message of length n packets to your friend. You know that at most k errors can occur during transmission. However, you are guaranteed that the first n–1 packets you send will arrive uncorrupted. How many packets must you send to guarantee successful transmission of your entire message? Why?

Answer

$n+2k$

We send the (n-1) packets that we know are safe.

We then send the last packet with 2k extra packets to protect for k general errors. Therefore, we need to send (n - 1) + (1 + 2k) packets.

Question [MT2 Fa14]

Secret Sharing with Spies (20 points)

An officer stored an important letter in her safe. In case she is killed in battle, she decides to share the password (which is a number) with her troops. However, everyone knows that there are 3 spies among the troops, but no one knows who they are except for the three spies themselves. The 3 spies can coordinate with each other and they will either lie and make people not able to open the safe, or will open the safe themselves if they can. Therefore, the officer would like a scheme to share the password that satisfies the following conditions:

- When *M* of them get together, they are guaranteed to be able to open the safe even if they have spies among them.
- The 3 spies must not be able to open the safe all by themselves.

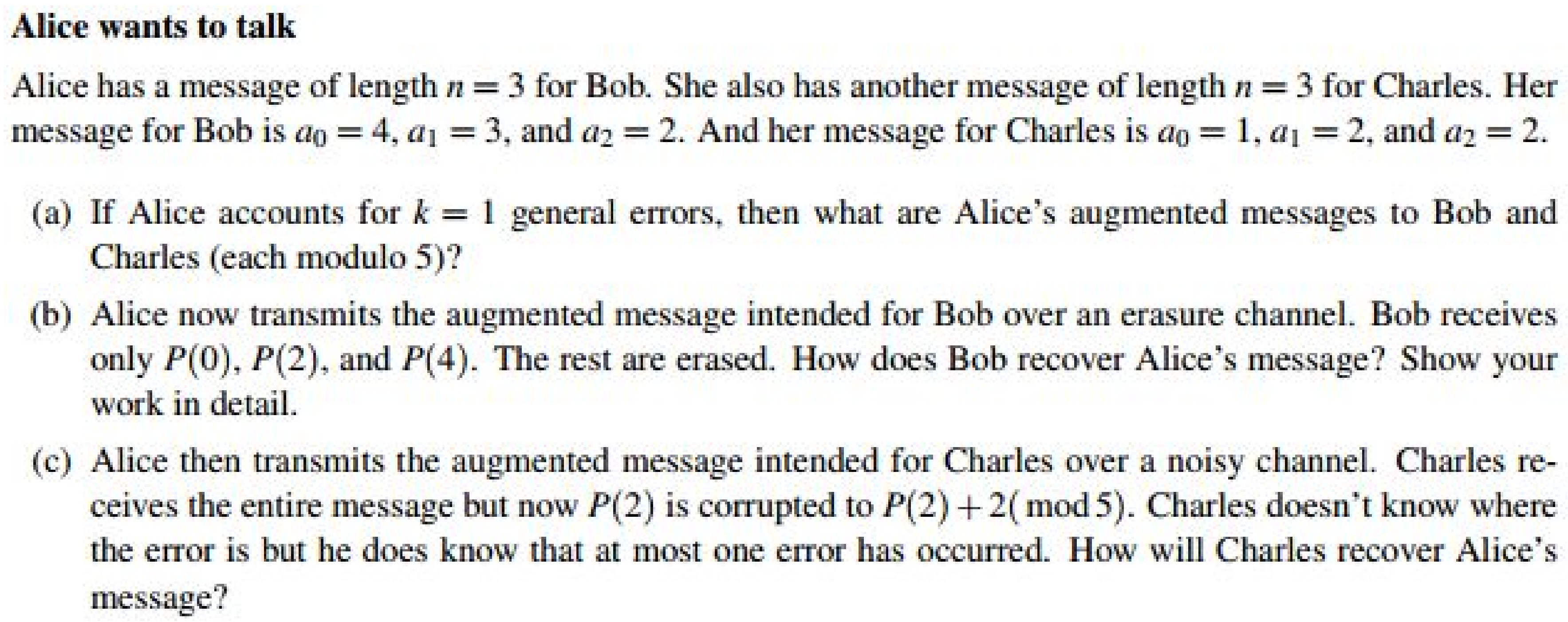
Please help the officer to design a scheme to share her password. What is the scheme? What is the smallest *M*? Show your work and argue why your scheme works and any smaller *M* couldn't work.

Answer

M = 10.

Think of the 3 spies as general errors! Also, the degree of the polynomial will have to be greater than or equal to 3. Otherwise, the spies can get the secret themselves.

Question [HW7 Fa14] - DIY!



Alice wants to talk

Alice has a message of length $n = 3$ for Bob. She also has another message of length $n = 3$ for Charles. Her message for Bob is $a_0 = 4$, $a_1 = 3$, and $a_2 = 2$. And her message for Charles is $a_0 = 1$, $a_1 = 2$, and $a_2 = 2$.

(a) If Alice accounts for $k = 1$ general errors, then what are Alice's augmented messages to Bob and Charles (each modulo 5)?

(b) Alice now transmits the augmented message intended for Bob over an erasure channel. Bob receives only $P(0)$, $P(2)$, and $P(4)$. The rest are erased. How does Bob recover Alice's message? Show your work in detail.

(c) Alice then transmits the augmented message intended for Charles over a noisy channel. Charles receives the entire message but now $P(2)$ is corrupted to $P(2) + 2 \pmod 5$. Charles doesn't know where the error is but he does know that at most one error has occurred. How will Charles recover Alice's message?

Answer

(see https://inst.eecs.berkeley.edu/~cs70/fa14/hw/sol7.pdf)

a. Using interpolation encoding:

   Sent to Bob: (4, 3, 2, 1, 0)

   Sent to Charles: (1, 2, 2, 1, 4)

b. (4, 3, 2)

c. (1, 2, 2)