Algorithms: motivation

Lecture 01.01

by Marina Barsky

http://barsky.ca/marina

So what is an algorithm?

So what is an algorithm?

Different definitions of an algorithm:

- *an unambiguous specification of how to solve a class of problems. Algorithms can perform calculation, data processing and automated reasoning* **Wikipedia**
- *a set of rules for solving a problem in a finite number of steps, as for finding the greatest common divisor* **Random House**
- *a procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation* **Merriam-Webster**
- *a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer* **Oxford**

A Russian stamp showing Persian mathematician Muhammad ibn Musa **Al-Khwarizmi** whose last name was transliterated to ***Algorithmi***

Algorithms?

Algorithm of happiness

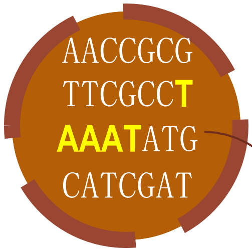
1. Focus upon problem-solving, not just venting
2. Build quality relationships with supportive people
3. Practice gratitude
4. Be kind to yourself, rather than overly self-critical
5. Set meaningful goals
6. Build intrinsic motivation

Algorithm of success in STEM classes

1. Break information into small chunks
2. Intensively concentrate on each chunk for at least 20 minutes
3. Take a break to let new material settle
4. Create a visual metaphor/story about each new concept
5. Solve problems several times until stable connection in your brain is formed

Reference

Algorithms in living systems (on biochemical hardware)



Code repository

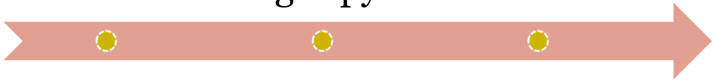
```
Algorithm  LIFE

X←input()

If X=”tiger”
    protein A=new Protein (TAAATA...)
```

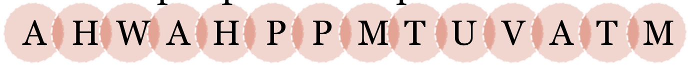
Program execution

Working copy of the code



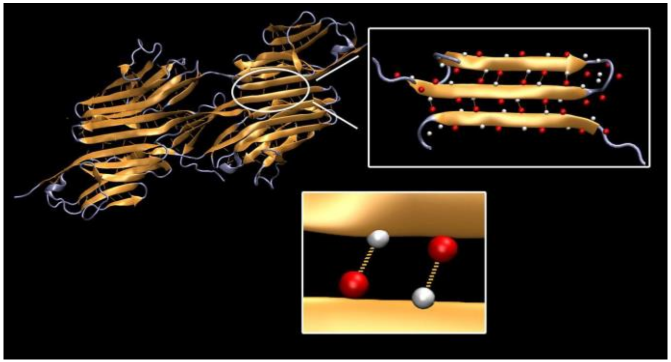
Sequence-dependent folding

Output protein sequence



Algorithmic assignments of the future

1. Implement an algorithm to create a living system which produces a biodegradable plastic
2. Implement a bacteria which produces spider silk
3. ...



```
Algorithm SPIDER_SILK
        (bacterial genome,
        silk protein sequence)
```

But in this course

- We limit ourselves to algorithms for performing computational tasks
- The algorithms should be very precise and unambiguous, so they can be communicated to a machine
- Every computational problem is an algorithmic problem

Sample problem: compute max value in list *t*

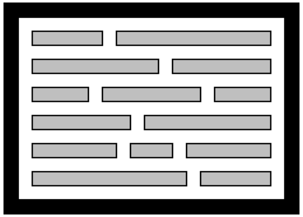
```
def array_max(t):
    max_val = None
    for x in t:
        if max_val is None or x > max_val:
            max_val = x

    return max_val
```

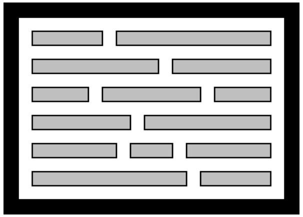
You were already using algorithms all the time!

Problems with straightforward natural solutions

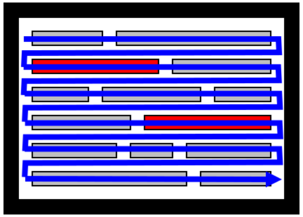
Display given text



Total number of words



Search for a given word



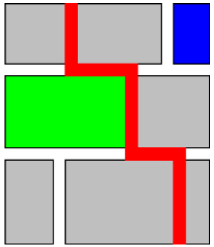
Linear scan

- Natural solution is already efficient (cannot do much better no matter what)

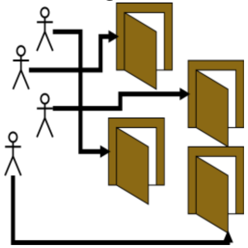
We assume that you already know how to solve such problems, and these algorithms are not covered in this course

Algorithmic problems

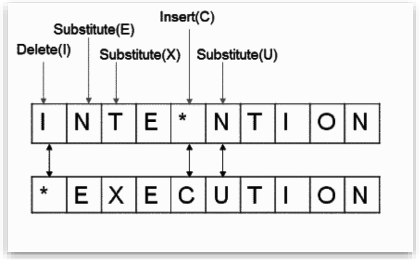
Shortest path



Best student-to-dorm assignment



Minimum number of operations to convert one word into another

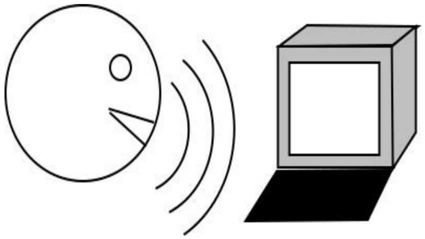


- Exhaustive search is not feasible
- Hard to do efficiently

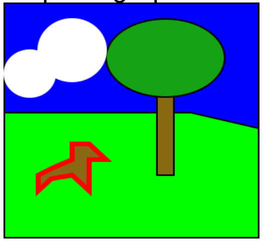
These are the problems we study in this course

AI problems

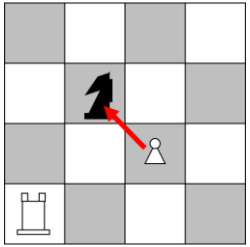
Understand natural language



Identify objects in photographs



Play games



- Require an additional step: **formally defining a computational problem**
- Contain multiple sub-algorithms and heuristics

These problems are covered in courses on AI/Machine Learning

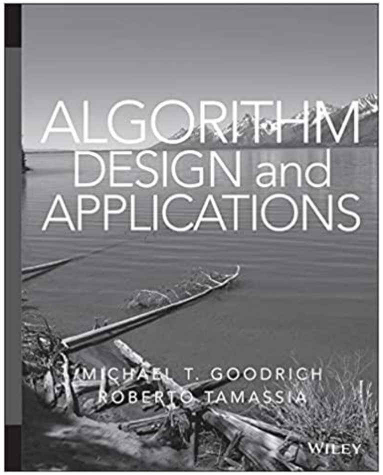
Course mechanics

Materials

The course textbook is **Algorithm Design and Applications, 2014, by Michael T. Goodrich, Roberto Tamassia. (ADA)**

You will be assigned selected readings.
The book also covers data structures, which you are already familiar with. We will review some of these next week.

We may have additional readings based on selected chapters from other books and online resources.



Implementing algorithms

"Learning algorithms without implementing them is not unlike learning surgery based solely on reading an anatomy book."

Donald Knuth

What language to use for algorithm implementation?

Considerations:

- C:
  C is a bare language with no built-in data structures - thus we have an opportunity to master the craft of building data structures from scratch. The language explicitly operates on raw bytes: we have full control over memory and instructions.

- Java/C++:
  You need to know main libraries, understand Object-Oriented paradigm, types, heap and stack. Java especially is a bit too verbose for algorithm implementation.

- Python:
  Clean and concise language, the closest to the pseudocode. We have no control over memory and sometimes we get puzzling artefacts that we cannot fix. But **for this course - Python is the best choice**.

Homework assignments

There are several problem sets – one per week – starting from week 3
Each set contains:

1. Problem-solving part which may include algorithm design, analysis, proofs and pseudocode.
2. Experimental algorithmics. Implementations. Experiments.

You are allowed to submit up to one week late, with 40% late submission penalty.

People who require accommodations need to set it up before the assignment is published.
They will get a one-week extension without penalty.

Exams and final grades

There will be two exams in the course.

- The Midterm Exam will be held online or in class before the spring break.
- The Final Exam will be held online or in person during the exam period.

The final grades will be computed as follows

- In-class Activities an Quizzes 20%
- Assignments 30%
- Midterm Exam 20%
- Final Exam* 30%

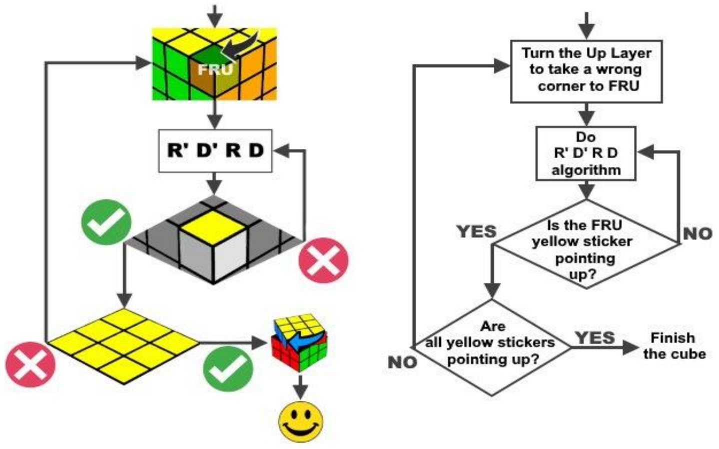
  Up to 5% extra for in-class participation and engagement

*You have to score at least 40% on the final exam in order to pass the course

Why study algorithms?

Why study algorithms?

- Mastery of algorithms is required for all branches of Computer Science: Cryptography? Networks? Graphics? Bioinformatics? AI?
- Algorithms play a key role in innovations of modern life
- Challenging yourself is good for the brain development
- Fun, addictive activity which can make you a better problem-solver in general


FRU
R' D' R D
Turn the Up Layer to take a wrong corner to FRU
Do R' D' R D algorithm
Is the FRU yellow sticker pointing up?
YES
NO
Are all yellow stickers pointing up?
YES
NO
Finish the cube

Rubik cube solving algorithm

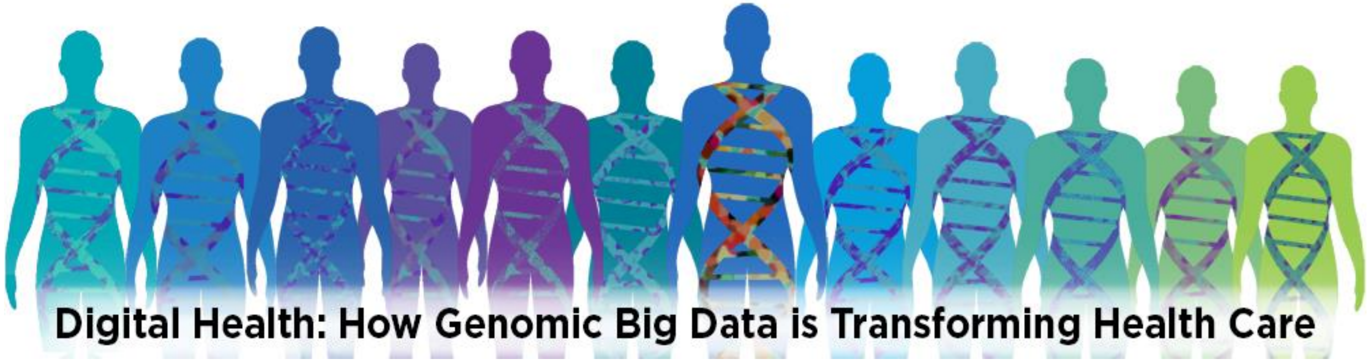
Programmers! you need efficient algorithms!

Memory and processing power constraints

- Ancient consoles
- Mobile devices
- Browsers

Ever more ambitious tasks

- Big data
- Machine learning
- DNA analysis

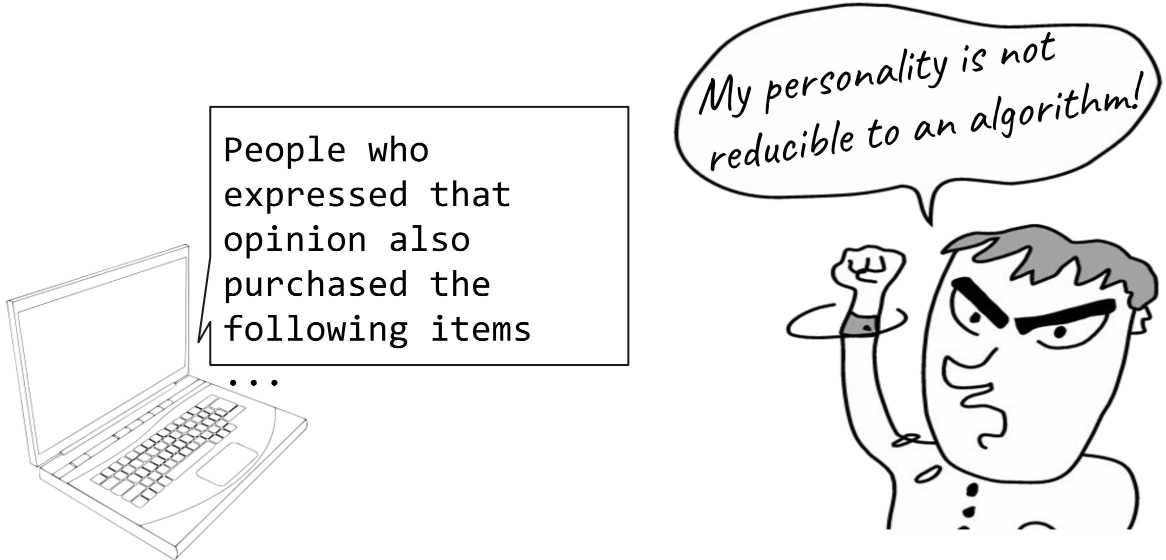

Digital Health: How Genomic Big Data is Transforming Health Care

Skills you will develop

- Become a better programmer
- Sharpen your mathematical and analytical skills
- Start “thinking algorithmically”
- Literacy with computer science’s “greatest hits”
- Ace your google interview?

Algorithms that changed modern world

- Google search: page-rank
- Online banking: concurrent transactions
- Online payments: public-key cryptography
- Reliable communication: error-correcting codes
- GPS systems: shortest paths

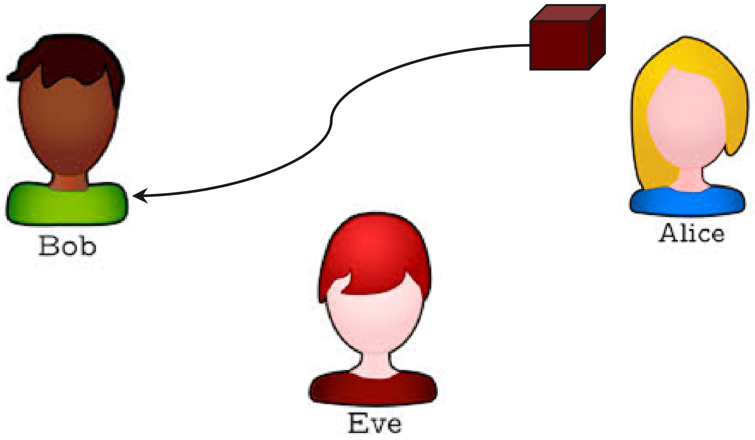

People who expressed that opinion also purchased the following items ...
My personality is not reducible to an algorithm!

Greatest Hits: RSA

Just for inspiration

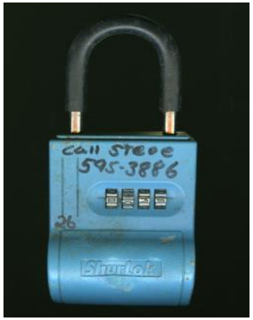
Problem: sending secret messages

- Alice wants to send a secret message to Bob - such that only Bob can read the message
- In order to make it possible, both Bob and Alice need to share an encryption/decryption key
- If only Bob and Alice know the key, Eve cannot decipher the secret message

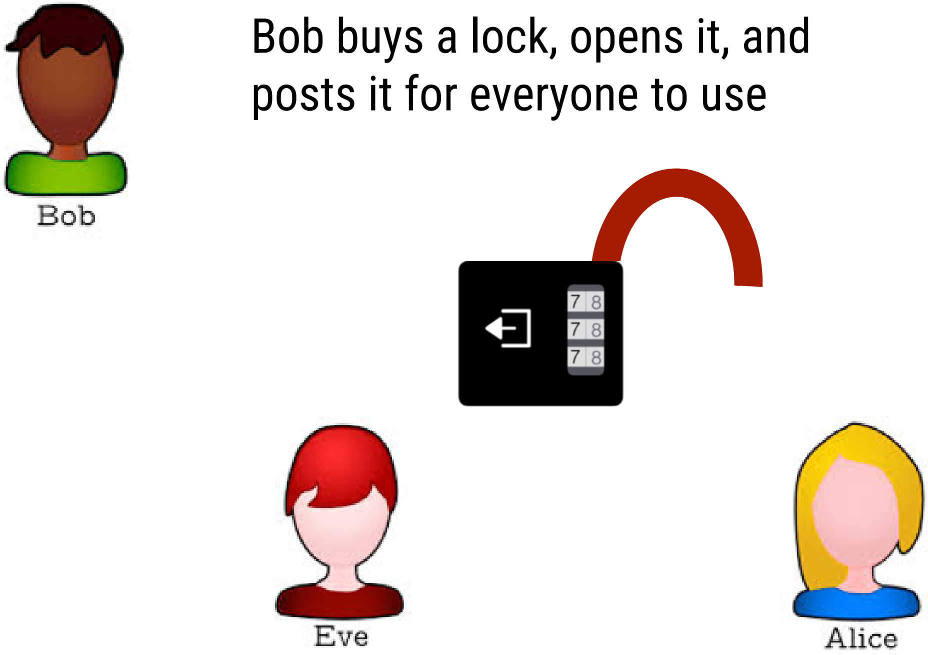


How to deliver a separate secret key to every user of your online store?

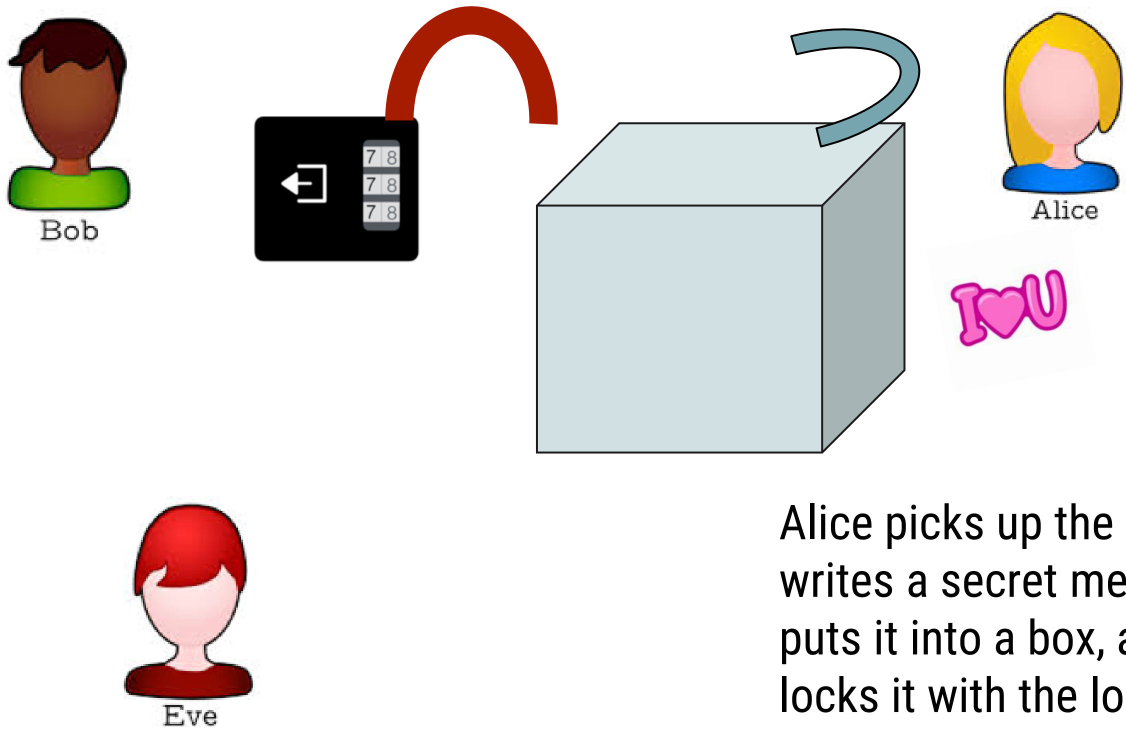
Raw idea: padlock



Bob buys a lock, opens it, and posts it for everyone to use

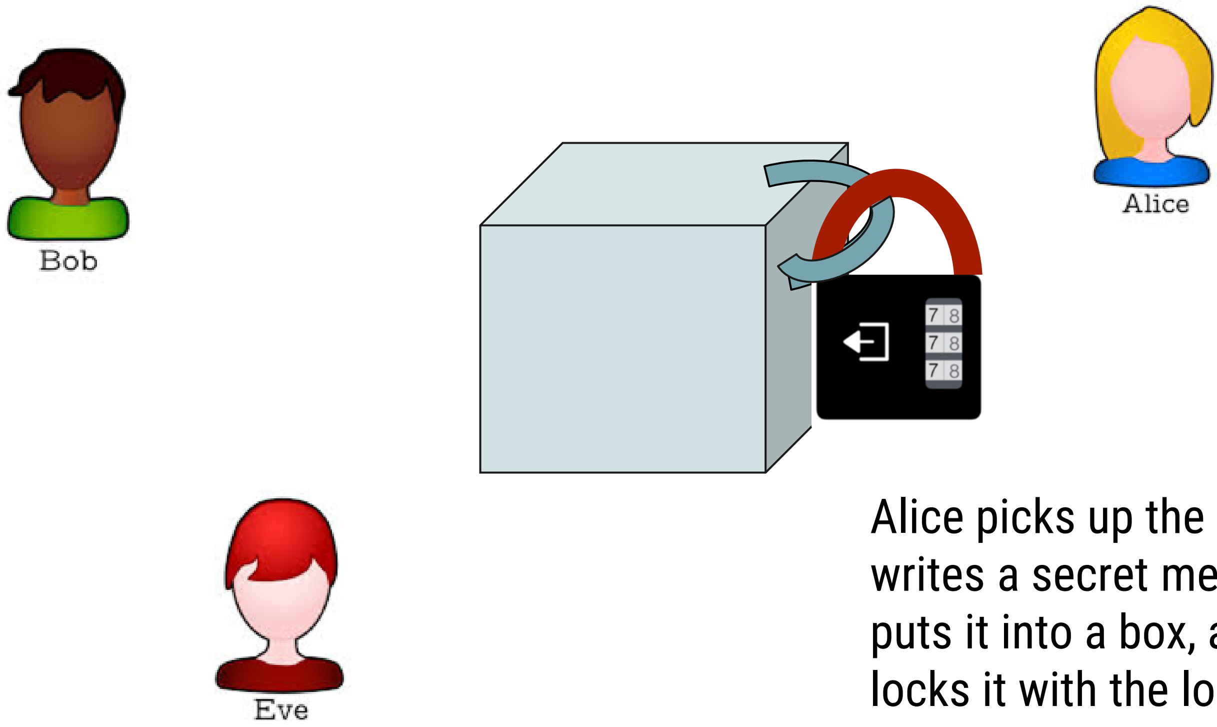


Public lock - private key



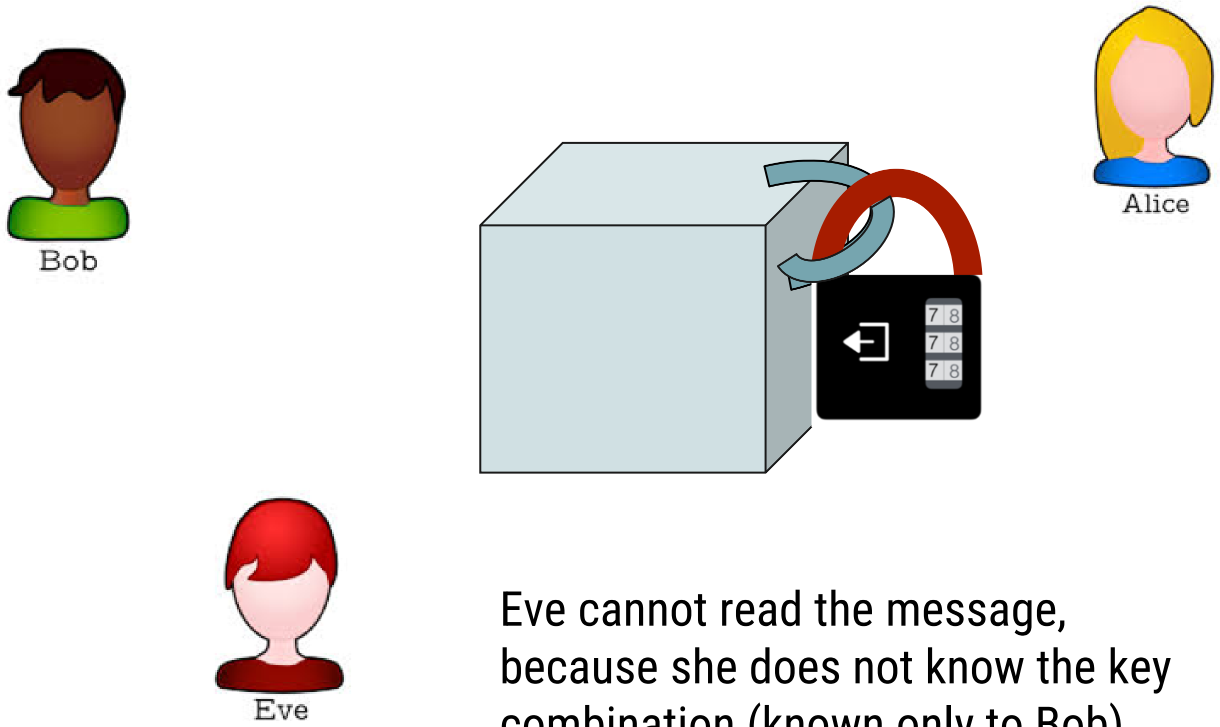
Alice picks up the lock, writes a secret message, puts it into a box, and locks it with the lock

Public lock - private key



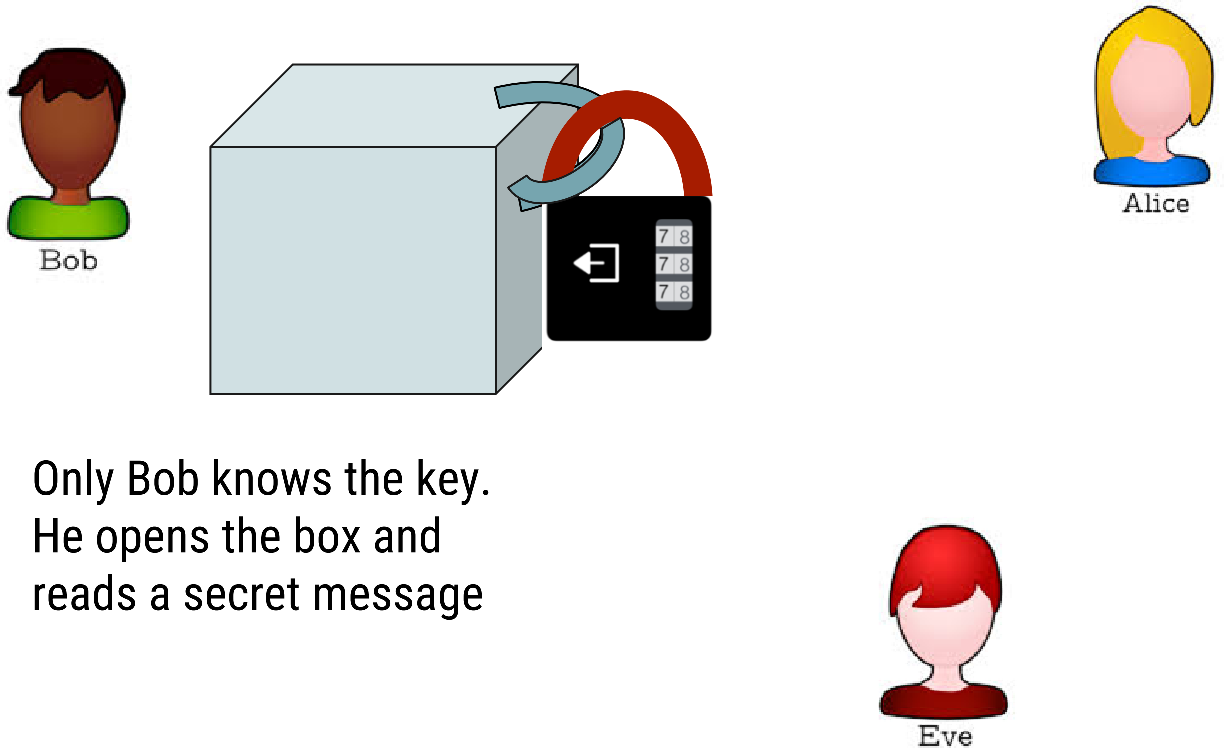
Alice picks up the lock, writes a secret message, puts it into a box, and locks it with the lock

Public lock - private key



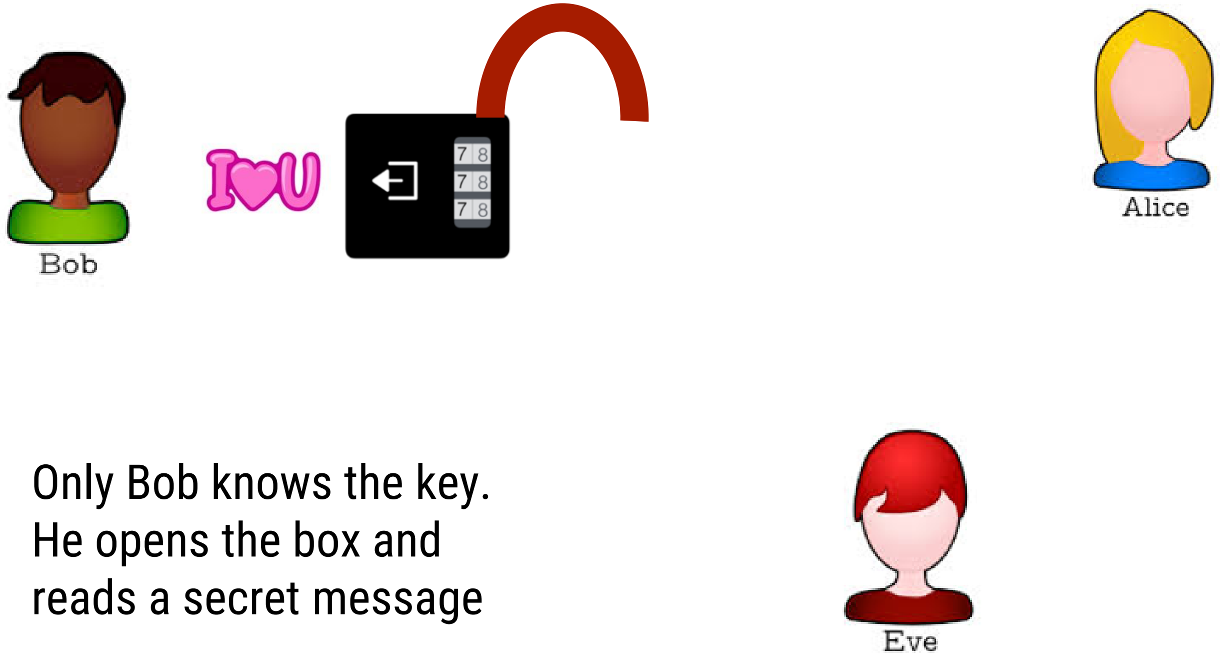
Eve cannot read the message, because she does not know the key combination (known only to Bob)

Public lock - private key



Only Bob knows the key.
He opens the box and
reads a secret message

Public lock - private key



Only Bob knows the key.
He opens the box and
reads a secret message

Did you see any physical open locks posted on the internet?

Idea: mathematical locks

- Bob generates a public key for everyone to use, and a private key that only Bob knows.

- The public key can be applied to each message with ease.
- But the decryption is impossible without knowing some additional information stored in the private key - and only Bob has this information.

Mathematical locks: hypothetical example

Imagine that we live in the world where **division by a number <10** is computationally easy, but **division by a number > 10** is prohibitively expensive - impossible to accomplish in 1000 years.

Mathematical locks: hypothetical example

division by a number <10 easy

division by a number > 10 impossible in 1000 years.

- Bob generates a **public encryption key** by multiplying two numbers 5*3=**15**. This is a public key, that he posts for everyone to use. Numbers **5** and **3** make the **private key** known only to Bob.

Mathematical locks: hypothetical example

division by a number <10 easy

division by a number > 10 impossible in 1000 years.

- Bob generates a **public encryption key** by multiplying two numbers 5*3=**15**. This is a public key, that he posts for everyone to use. Numbers **5** and **3** make the **private key** known only to Bob.
- Alice wants to send a secret number **4**. She multiplies it by 15 and sends **60**. Nobody (including Alice and Bob) knows an efficient algorithm of dividing 60 by 15.

Mathematical locks: hypothetical example

division by a number <10 easy

division by a number > 10 impossible in 1000 years.

- Bob generates a **public encryption key** by multiplying two numbers 5*3=**15**. This is a public key, that he posts for everyone to use. Numbers **5** and **3** make the **private key** known only to Bob.
- Alice wants to send a secret number **4**. She multiplies it by 15 and sends **60**. Nobody (including Alice and Bob) knows an efficient algorithm of dividing 60 by 15.
- But Bob knows that 15 is nothing else but 3*5. He knows it because he created the key. Bob gets the message:
  60 / **3** = 20, 20 / **5** = **4**

Mathematical locks: hypothetical example

division by a number <10 easy

division by a number > 10 impossible in 1000 years.

- Bob generates a **public encryption key** by multiplying two numbers 5*3=**15**. This is a public key, that he posts for everyone to use. Numbers **5** and **3** make the **private key** known only to Bob.
- Alice wants to send a secret number **4**. She multiplies it by 15 and sends **60**. Nobody (including Alice and Bob) knows an efficient algorithm of dividing 60 by 15.
- But Bob knows that 15 is nothing else but 3*5. He knows it because he created the key. Bob gets the message:
  60 / **3** = 20, 20 / **5** = **4**

The only problem to solve: to find a real function that works this way!

RSA encryption

(**R**ivest, **S**hamir, **A**dleman, since 2000)

The RSA encryption idea is based on **prime number factorization**

https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/intro-to-rsa-encryption

http://doctrina.org/How-RSA-Works-With-Examples.html

Prime factorization

Euclid: every number has only one prime factorization:

14 = 2*7

24 = 2*2*2*3

Prime factorization of large numbers

Euclid: every number has only one prime factorization:

14 = 2*7

24 = 2*2*2*3

What is the prime factorization of the following 100-digit number?
9,412,343,607,359,262,946,971,172,136,294,514,357,528,981,378,98
3,082,541,347,532,211,942,640,121,301,590,698,634,089,
611,468,911,681

Prime factorization of large numbers

What is the prime factorization of the following 100-digit number?
9,412,343,607,359,262,946,971,172,136,294,514,357,528,981,378,98
3,082,541,347,532,211,942,640,121,301,590,698,634,089,
611,468,911,681

It is:

86,759,222,313,428,390,812,218,077,095,850,708,048, 977
*
108,488,104,853,637,470,612,961,399,842,972,948,409,834,611,525,
790,577,216,753

- There are no other factors: these two factors are **prime**
- Finding them is quite a job: in fact, it's a several-month project for a supercomputer

Totient function

Euler: *totient function* φ(n) (/faɪ/)
counts how many integers smaller than *n* have the Greatest Common Divisor (GCD) with *n* equal to 1 (have no common divisors except 1).
Two numbers *m* and *n* for which GCD(*m*,*n*) = 1 are called *co-primes*

n=8
1,2,3,4,5,6,7 $\longrightarrow$ φ(8) = 4

Totient function of a prime number

Euler: *totient function* $\varphi(n)$ (/faɪ/)
counts how many integers smaller than *n* have the Greatest Common Divisor (GCD) with *n* equal to 1.

If the number *n* is a prime, then $\varphi(n) = n - 1$
$n=11 \rightarrow$ all $n-1$ are co-primes: 1,2,3,4,5,6,7,8,9,10

Euler proved that:
$\varphi(n*m) = \varphi(n)*\varphi(m)$, if both *n* and *m* are distinct primes

3	5	15
1,2	1,2,3,4	1,2,3,4,5,6,7,8,9,10,11,12,13,14
$\varphi(3)=2$	$\varphi(5)=4$	

Totient function of a prime number

Euler: *totient function* φ(n) (/faɪ/)

counts how many integers smaller than n have the Greatest Common Divisor (GCD) with n equal to 1.

If the number n is a prime, then φ(n) = n - 1

n=11 → all n-1 are co-primes: 1,2,3,4,5,6,7,8,9,10

Euler proved that:

φ(n*m) = φ(n)*φ(m), if both n and m are distinct primes

3	5	15
1,2	1,2,3,4	1,2,3,4,5,6,7,8,9,10,11,12,13,14
φ(3)=2	φ(5)=4	φ(15)=8

Euler theorem

Euler theorem:

For any two numbers *n*, *x* which are ***co-prime*** (do not share common divisors):

$$\mathbf{x^{\varphi(n)} \equiv 1 \pmod{n}}$$

Meaning: $x^{\varphi(n)} \bmod n = 1$

($x^{\varphi(n)}$ divided by n gives a remainder 1)

Euler theorem: examples

Euler theorem:

For any two numbers n, x which are ***co-prime***:

$x^{\varphi(n)} \bmod n = 1$

For example let's say n=8

1,2,3,4,5,6,7

Then $\varphi(n) = 4$

Which number is a co-prime of 8?

Let x=3. $3^4 \bmod 8 = 81 \bmod 8 = 1$

Let x=9. $9^4 \bmod 8 = 6561 \bmod 8 = (820*8 + 1) \bmod 8 = 1$

Euler theorem: k times

Euler theorem:

For any two numbers n, x which are ***co-prime*** (do not share common divisors):

$\mathbf{x^{\varphi(n)} \equiv 1 \pmod{n}}$ (1)

Meaning: $x^{\varphi(n)} \bmod n = 1$

($x^{\varphi(n)}$ divided by n gives a remainder 1)

If we multiply both sides of equation (1) by itself k times we get:

$\mathbf{x^{k\varphi(n)} \equiv 1^k \pmod{n}}$

which means:

$x^{k\varphi(n)} \bmod n = 1$ (if x is co-prime of n then so is x^k)

We can get x back from $x^{\varphi(n)}$

Euler theorem:

For any two numbers n, x which are ***co-prime***:

$x^{k\varphi(n)} \bmod n = 1$

$x^{k\varphi(n)} \bmod n = 1$

$\underline{{*}x \text{ (multiply both sides by x)}}$

$x * (x^{k\varphi(n)} \bmod n) = x$

Meaning: remainder from division of $x^{\varphi(n)+1}$ by n is x!

$x^{k\varphi(n)+1} \bmod n = x$

RSA: encryption

Euler theorem:

For any two numbers n, x which are ***co-prime***:

$x^{k\varphi(n)+1} \bmod n = x$

Say we create **2 public keys**: **e** and **n**

Then to encrypt the secret number **x** use:

$x^{e} \bmod n \longrightarrow y$

RSA: decryption

Euler theorem:

For any two numbers n, x which are ***co-prime***:

$\mathbf{x^{k\varphi(n)+1} \bmod n = x}$

To encrypt the secret number **x** we used:

$\mathbf{x}^{\mathbf{e}} \bmod \mathbf{n} \longrightarrow \mathbf{y}$

To get x back from y use Euler theorem:

$y^{(k\varphi(n)+1)/e} \bmod n \longrightarrow \mathbf{x}$

The decryption exponent **d** is computed as $\mathbf{d=(k\varphi(n)+1)/e}$

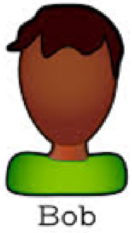
This can be done easily - **but only if we know φ(n)**

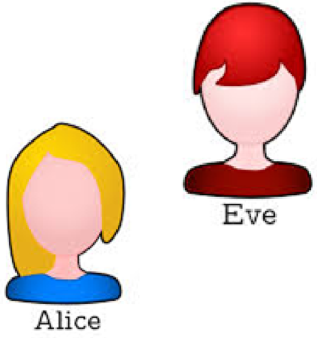
RSA: why we cannot break it

The decryption exponent is computed as $d=(k\varphi(n)+1)/e$
This can be done easily - but only if we know $\varphi(n)$

- Let n be a multiplication of 2 100-digit primes p_1 and p_2
- If n is a multiplication of 2 large primes, then any small integer will be a co-prime of n
- Then everyone can compute the decryption exponent easily, but only if they know the original primes: $\varphi(n) = (p_1-1)(p_2-1)$
- If they do not know the original primes, it is not feasible to compute function $\varphi(n)$ just by knowing n.
- Because it involves either computing prime factorization of n, or common divisors of n with every number < n. Both are computationally hard.

RSA example: Bob wants to receive secrets

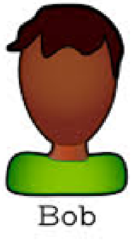


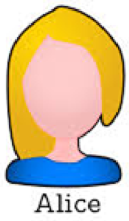


- Bob picks 2 “large” prime numbers $p_1 = 3$ and $p_2 = 5$
- Generates a **public** key: **n**=3*5 = **15**
- Computes ***φ(n)*** = (***3***−1)(***5***−1) = 8 and keeps it to himself
- Chooses encryption exponent **e** to be a random prime number less than φ that is also not a divisor of φ
  k*$\varphi(n)$+1 should be divisible by e → chooses **e=3** (k=1)
- Computes decryption exponent **d** = (1φ(n)+1)/e, (8+1)/3 = **3** and keeps it **private**

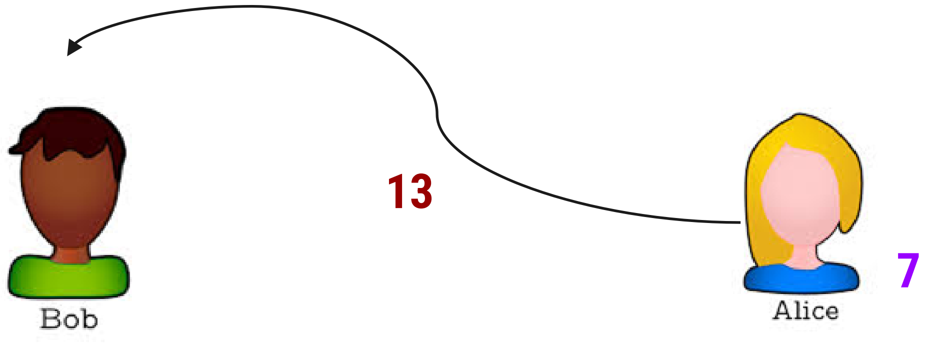
RSA example: public key

Bob posts **$n=15$**, **$e=3$**
as a public lock:
$x^e \bmod n$





RSA example: public encryption



Only Bob knows decryption exponent **d** = 3

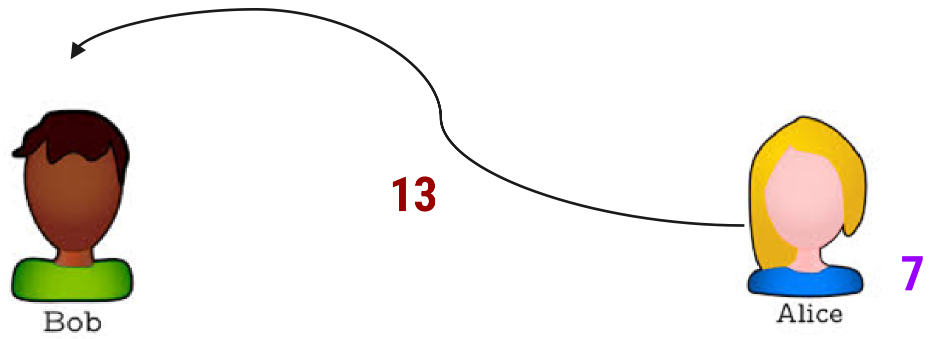
n=15, **e=3**

Alice wants to send number **7**

She sends encrypted
$y = 7^3 \bmod 15 =$ **13**

power mod calculator

RSA example: message is safe

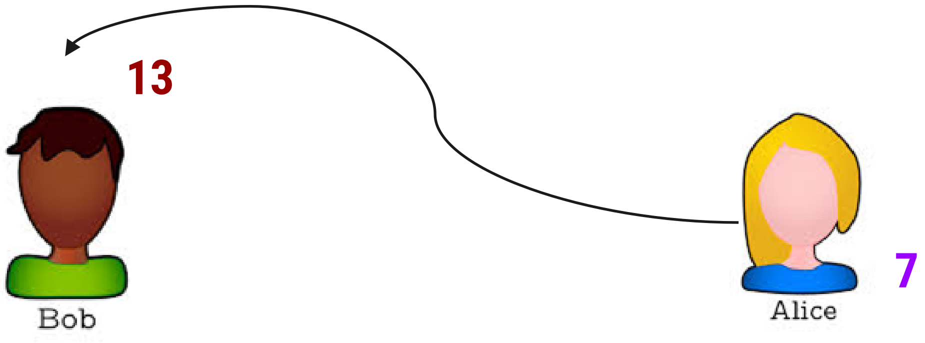


n=15, **e=3**

y = 13 circulates over the internet in open

Eve has no way of computing an original message x from y

RSA example: decrypting the message



n=15, **e=3**

Bob receives y = 13 and decrypts it using private key **d**=3:

$x = y^d \bmod n = 13^3 \bmod 15 = 7$

Class activity: RSA experience

Power mod calculator:
https://www.mtholyoke.edu/courses/quenell/s2003/ma139/js/powermod.html

What we learned from this experience

- Intuitive ideas are often simple (padlock)
- Every physical idea can be made abstract with the help of math (finding asymmetrical function which is easy to do one way but hard to reverse)
- Computer is a tool that allows us materialize the most abstract ideas and change the real life

RSA encryption: summary

- All arithmetic is done modulo n, with n=pq where p and q are large primes.
- Decryption in this system relies on computing Euler's *phi* function, φ(n), which is hard to compute (hence the system is hard to break) unless you know the prime factorization of n (which is also hard to compute unless you know it upfront)

$x^e \bmod n = y$ hard to reverse: $x^e = y + kn$

Easy to recover $x = y^d \bmod n$, where $d = (k\varphi(n)+1)/e$

- Do not implement RSA by yourself: use a library
- But finding if two numbers are co-primes? Prime factorization? We can try

“If I have perchance omitted anything more or less proper or necessary, I beg indulgence, since there is no one who is blameless and utterly provident in all things.”

Leonardo of Pisa a.k.a. Fibonacci (1170–c1250),
a remarkable Italian mathematician,
the author of Liber Abaci (“The Book of Calculation”),
one of the most consequential mathematical books in history

To do list

- Have fun with RSA (optional)
- Find the main textbook
- Brush up your Python skills
- Review course mechanics and prepare your questions for the next meeting