Lecture 28: Computer Security

Brian Hou
August 9, 2016

Many slides are adapted from CS 161 (Computer Security)

Announcements

- Final Exam on Friday (8/12) from 5–8pm in 155 Dwinelle
- Scheme Recursive Art submissions due today (8/9)!
- Potluck II tomorrow (8/10)! 5–8pm in Wozniak Lounge
- Homework 10 is due today (8/9)
  - AutoStyle EC portion due 8/10, last part due 8/11
- Homework 11 and 12 will be due 8/10 and 8/12
  - Last two of the three extra credit surveys
  - Vote for your favorite Recursive Art submissions!
- Check your grades! Details on Piazza, regrades close 8/10

Roadmap

- Introduction
- Functions
- Data
- Mutability
- Objects
- Interpretation
- Paradigms
- Applications

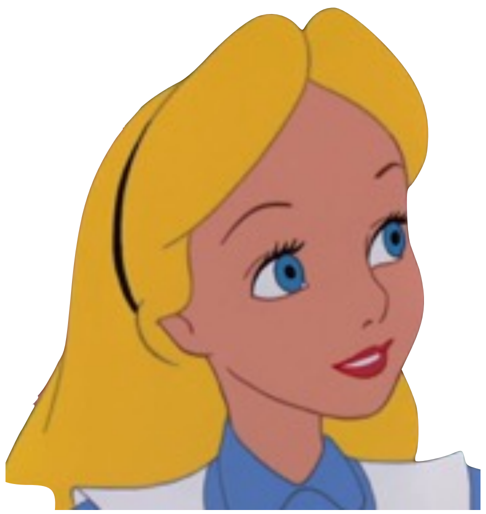
- This week (Applications), the goals are:
  - To go beyond CS 61A and see examples of what comes next
  - To wrap up CS 61A!

Computer Security

Computer Security

- A subfield of computer science with two main goals:
  - Allow intended use of computer systems
  - Prevent unwanted use that may cause harm
- Why should you care?
  - The Internet has a lot of information about you...
- Today, we'll look at two problems:
  - Cryptography: secure communication over insecure communication channels
  - Injection Attacks

Today's Special Guests!



Alice

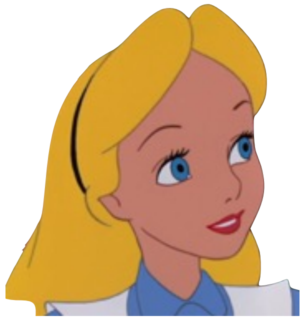
Bob

The Adversary
(Eve or Mallory)

Cryptography

Cryptography

- Three main goals: confidentiality, integrity, authenticity
- Today, we'll focus on confidentiality
- *Confidentiality*: prevent adversaries from reading private communications
  - Can Alice and Bob communicate in a way that even an eavesdropper Eve can't understand what they're saying?



The Caesar Cipher (demo)

- One of the first attempts to encrypt a message
  - Was used by Roman dictator Julius Caesar
- Alice and Bob agree on a secret number (*key*) between 0 and 25 to shift the alphabet
  - For example, if the number is 2 then 'A' becomes 'C', 'B' becomes 'D', ..., 'Y' becomes 'A', 'Z' becomes 'B'

http://www.cryptoclub.org/tools/caesar_cipher.php

Breaking the Caesar Cipher (demo)

```
vgg ocz rjmgy'n v novbz ,
viy vgg ocz hzi viy rjhzi hzmzgt kgvtzmn :
oczt cvqz oczdm zsdon viy oczdm ziomvixzn ;
viy jiz hvi di cdn odhz kgvtn hvit kvmon ,
```

- Observation: There are only 26 possible keys
- Observation: Computers are fast
- Observation: Letters don't appear in English with the exact same frequency
  - For example, 'E' appears more often than 'Z'

The Enigma Machine

- Used by the German military in World War II
- First broken by Polish mathematicians in 1932
- Information gained by the Allied forces is estimated to have shortened fighting by two years
- Implemented a progressive substitution cipher (e.g. different shift for each letter of the message)

Better Cryptography

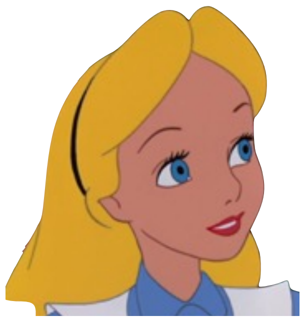
- This will require a bit of math, but the detailed steps aren't particularly important
- From here onward, we'll represent a message with a number **m**, rather than a string of characters
- Main idea: It is feasible to find three large numbers **e**, **d**, and **n** such that $(m^e)^d = m \pmod{n}$

The RSA Algorithm

- RSA is an example of *public-key cryptography*
  - The public key is known to everyone and is used to encrypt messages for the user
  - The private key is only known by the user and is the only way to decrypt a message
  - This is also known as *asymmetric cryptography*: the message sender and recipient have two different keys
- Main idea: It is feasible to find three large numbers **e, d,** and **n** such that $(m^e)^d = m \pmod{n}$
- Public key: **e** and **n** ("modulus")
- Private key: **d**

RSA Encryption and Decryption

- Suppose that Bob wants to send a message **m** to Alice
- He can encrypt a message by computing $\mathbf{c = m^e \pmod{n}}$
  - Everyone knows that Alice's public key is **e** and **n**
- She can decrypt his message by computing $\mathbf{c^d = (m^e)^d = m \pmod{n}}$
  - Only Alice knows her private key **d**



Breaking RSA

- Eve needs to compute **d** to decrypt the message
- **e, d,** and **n** aren't just three arbitrarily chosen numbers!
  - **$n = pq$,** where **p** and **q** are two very large primes ($\sim 2^{1024}$)
  - For RSA encryption and decryption to work**, $ed = 1 \pmod{(p-1)*(q-1)}$** (Euler's totient theorem)
- As far as we know, computing **d** means that we have to
  1. Factor **n** into **p** and **q**
  2. Solve **$ed = 1 \pmod{(p-1)*(q-1)}$** for **d**
- It turns out that Step 2 is easy and Step 1 is hard!
- The security of RSA relies on factoring being difficult

Factoring is (Maybe) Hard

- Quick! Factor 561!
- There is no known efficient factoring algorithm
- Researchers spent 2007–2009 on factoring a 768-bit modulus (232 digits)
  - It took the equivalent of almost 2000 years of computing
  - Factoring a 1024-bit RSA modulus would be 1000x harder, but could happen in the next decade (2019 is coming up!)

Factoring Complexity

- When people talk about factoring complexity, they typically describe runtime with respect to the bits that it takes to represent the number n (i.e. $\log_2 n$)
- Factoring is in NP: the answer can be verified by multiplying, which takes polynomial time
- We don't know if factoring is in P: the best algorithms for factoring are better than exponential but worse than polynomial
- Quantum computers can factor large numbers in polynomial time with Shor's algorithm
  - But their most recent breakthrough was factoring 21, so...

Applications of RSA

- For now (and for many years to come), RSA is secure
- Many protocols rely on RSA today
  - SSH (how to connect securely to the lab computers)
  - SSL/TLS (the "S" in "HTTPS", how to connect securely to Facebook, etc.)

Break!

Injection Attacks

Compromising Web Servers

- What could you do if you controlled one of Facebook's servers?
- Steal sensitive data (e.g. data from many users)
- Change server data (e.g. affect users)
- Gateway to enabling attacks on users
- Impersonation (of users to servers, or vice versa)

Code Injection Attacks (demo)

- Injection attacks are one way to compromise web servers
- People first started talking about this back in 1998, with hundreds of proposed fixes and solutions
- General attack structure:
  - Attacker user provides some bad input
  - Web server does not check input format
  - Enables attacker to execute arbitrary code on the server

Summary

- Computer security studies how we can allow for the intended use of computer systems while preventing unwanted use that may cause harm
- Cryptography studies how we can communicate with others securely
- As programmers, we must be mindful of security best practices when developing applications
  - Even then, it might not be enough!
- CS 161 (Computer Security) goes into much more depth
- CS 261 and CS 276 are the graduate-level security and cryptography classes