Week 4 - Friday

COMP 4290

Last time

- What did we talk about last time?
- Finished AES
- Public key cryptography
- Started number theory

Questions?

Project 1

Spencer Wilson Presents

More Number Theory!

Greatest common divisor

- The greatest common divisor or GCD of two numbers gives the largest factor they have in common
- Example:
  - GCD(12, 18) =
  - GCD(42, 56) =
- For small numbers, we can determine GCD by doing a complete factorization

Euclid's algorithm

- For large numbers, we can use Euclid's algorithm to determine the GCD of two numbers
- Algorithm GCD(***a***, ***b***)
  1. If ***b*** = 0
     - Return ***a***
  2. Else
     - ***temp*** = ***a*** mod ***b***
     - ***a*** = ***b***
     - ***b*** = ***temp***
  3. Goto Step 1
- Example: GCD(1970, 1066)

Extended Euclid's algorithm

- We can extend Euclid's algorithm to give us the multiplicative inverse for modular arithmetic
- Example: Find the inverse of 120 mod 23
- Let ***a*** be the number
- Let ***b*** be the modular base

```
Find Inverse(a, b)
    x = 0
    lastx = 1
    y = 1
    lasty = 0
    while b ≠ 0
        quotient = a div b
        temp = b
        b = a mod b
        a = temp
        temp = x
        x = lastx – quotient * x
        lastx = temp
        temp = y
        y = lasty – quotient * y
        lasty = temp
    Return lastx
```

Fermat's Little Theorem

- If ***p*** is prime and ***a*** is a positive integer not divisible by ***p***, then:

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

Proof of Fermat's Theorem

- Assume a is positive and less than p
- Consider the sequence $a, 2a, 3a, \ldots, (p-1)a$
- If these are taken mod p, we will get (in a different order):
  - $1, 2, 3, \ldots, p-1$
  - This bit is the least obvious part of the proof
  - However (because p is prime) if you add any non-zero element repeatedly, you will eventually get back to the starting point, covering all values (except 0) once
- Multiplying this sequence together gives:
  - $a \cdot 2a \cdot 3a \cdot \ldots \cdot (p-1)a \equiv 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1) \pmod{p}$
  - $a^{p-1}(p-1)! \equiv (p-1)! \pmod{p}$
  - $a^{p-1} \equiv 1 \pmod{p}$

Euler's in the mix too

- Euler's totient function is written $\phi(\boldsymbol{n})$
- $\phi(\boldsymbol{n})$ = the number of positive integers less than ***n*** and relatively prime to ***n*** (including 1)
- If ***p*** is prime, then $\phi(\boldsymbol{p}) = \boldsymbol{p} - 1$
- If we have two primes ***p*** and ***q*** (which are different), then: $\phi(\boldsymbol{pq}) = \phi(\boldsymbol{p})\cdot\phi(\boldsymbol{q}) = (\boldsymbol{p} - 1)(\boldsymbol{q} - 1)$

Take that, Fermat

- **Euler's Theorem:**
  For every $\boldsymbol{a}$ and $\boldsymbol{n}$ that are relatively prime,

  $$\boldsymbol{a}^{\phi(\boldsymbol{n})} \equiv 1 \pmod{\boldsymbol{n}}$$

- This generalizes Fermat's Theorem because $\phi(\boldsymbol{p}) = \boldsymbol{p} - 1$ if $\boldsymbol{p}$ is prime
- Proof is messier

RSA

RSA Algorithm

- Named for **R**ivest, **S**hamir, and **A**dleman
- Take a plaintext ***M*** converted to an integer

- Create a ciphertext ***C*** as follows:
  $C = M^e \bmod n$

- Decrypt ***C*** back into ***M*** as follows:
  $M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

The pieces

Term	Details	Source
M	Message to be encrypted	Sender
C	Encrypted message	Computed by sender
n	Modulus, $n = pq$	Known by everyone
p	Prime number	Known by receiver
q	Prime number	Known by receiver
e	Encryption exponent	Known by everyone
d	Decryption exponent	Computed by receiver
$\phi(n)$	Totient of n	Known by receiver

How it works

- To encrypt:
  $\boldsymbol{C} = \boldsymbol{M}^{\boldsymbol{e}} \bmod \boldsymbol{n}$
- ***e*** could be 3 and is often 65537, but is always publically known
- To decrypt:
  $\boldsymbol{M} = \boldsymbol{C}^{\boldsymbol{d}} \bmod \boldsymbol{n} = \boldsymbol{M}^{\boldsymbol{ed}} \bmod \boldsymbol{n}$
- We get ***d*** by finding the multiplicative inverse of ***e*** mod $\phi(\boldsymbol{n})$
- So, $\boldsymbol{ed} \equiv 1 \pmod{\phi(\boldsymbol{n})}$

Why it works

- We know that $\boldsymbol{ed} \equiv 1 \pmod{\phi(\boldsymbol{n})}$
- This means that $\boldsymbol{ed} = \boldsymbol{k}\phi(\boldsymbol{n}) + 1$ for some nonnegative integer $\boldsymbol{k}$
- $\boldsymbol{M}^{\boldsymbol{ed}} = \boldsymbol{M}^{\boldsymbol{k}\phi(\boldsymbol{n}) + 1} \equiv \boldsymbol{M} \cdot (\boldsymbol{M}^{\phi(\boldsymbol{n})})^{k} \pmod{n}$
- By Euler's Theorem
  $\boldsymbol{M}^{\phi(\boldsymbol{n})} \equiv 1 \pmod{\boldsymbol{n}}$
- So, $\boldsymbol{M} \cdot (\boldsymbol{M}^{\phi(\boldsymbol{n})})^{k} \equiv \boldsymbol{M} \pmod{\boldsymbol{n}}$

An example

- $M = 26$
- $p = 17$, $q = 11$, $n = 187$, $e = 3$
- $C = M^3 \bmod 187 = 185$
- $\phi(n) = (p - 1)(q - 1) = 160$
- $d = e^{-1} \bmod 160 = 107$
- $C^d = 185^{107} \bmod 187 = 26$
- If you can trust my modular arithmetic

Why it's safe

- You can't compute the multiplicative inverse of e mod $\phi(n)$ unless you know what $\phi(n)$ is
- If you know p and q, finding $\phi(n)$ is easy
- Finding $\phi(n)$ is equivalent to finding p and q by factoring n
- No one knows an efficient way to factor a large composite number
  - Or they're not telling

Future risks

- Public key cryptography would come crashing down if
  - Advances in number theory could make RSA easy to break
  - Quantum computers could make it easy to factor large composites

Practical considerations

- Choose your primes carefully
  - $p < q < 2p$
  - But, the primes can't be too close together either
  - Some standards insist that p and q are **strong primes**, meaning that $p - 1 = 2m$ and $p + 1 = 2n$ where m and n have large prime factors
  - There are ways to factor poorly chosen pairs of primes
- Pad your data carefully
- Take the example of a credit card number
  - If you know a credit card number is encrypted using RSA using a public n and an e of 3, how do you discover the credit card number?

Upcoming

Next time…

- Key management
- Hash functions
- Colm Oneacre presents

Reminders

- **Office hours today start late:**
  - **2:30-5 instead of 1:45-4**
- Keep reading 12.4
- Work on Project 1
  - **Due tonight!**