MA/CSSE 473 Day 11

Data Encryption

MA/CSSE 473 Day 11

- HW 5 is due tomorrow.
- HW 6 due Friday
- Poll results:

Result Summary		
12	55%	NO: Keep it T-F
10	45%	YES: Change to M-R

- Tomorrow I hope to discuss the Knuth interview, and Brute Force Algorithms
- **Student Questions**
- Cryptography Introduction
- RSA
- (If time) Amortized analysis, growable array review.

Cryptography

- I want to transmit a message **m** to you
  - in a form e(**m**) that you can readily decode by running d(e(**m**)),
  - That an eavesdropper has little chance of decoding
- Private-key protocols
  - You and I meet beforehand and agree on e and d.
- Public-key protocols
  - You publish an e for which you know the d, but it is very difficult for someone else to guess the d.
  - Then I can use e to encode messages that only you* can decode

* and anyone else who can figure out what d is if they know e.

Messages can be integers

- Since a message is a sequence of bits …
- We can consider the message to be a sequence of b-bit integers (where b is fairly large), and encode each of those integers.
- Here we focus on encoding and decoding a single integer.

RSA Public-key Cryptography

- Rivest-Shamir-Adleman (1977)
  - A reference : Mark Weiss, Data Structures and Problem Solving Using Java, Section 7.4
- Consider a message to be a number modulo N, an n-bit number (longer messages can be broken up into n-bit pieces)
- The encryption function will be a bijection on {0, 1, …, N-1}, and the decryption function will be its inverse
- How to pick the N and the bijection?

bijection: a function f from a set X to a set Y with the property that for every y in Y, there is exactly one x in X such that f(x) = y. In other words, f is both one-to-one and onto.

N = p q

- Pick two large primes, p and q, and let N = pq.
- **Property**: If e is any number that is relatively prime to N' = (p-1)(q-1), then
  - the mapping $x \rightarrow x^e \bmod N$ is a bijection on {0, 1, …, N-1}
  - If d is the inverse of e mod (p-1)(q-1), then for all x in {0, 1, …, N-1}, $(x^e)^d \equiv x \pmod N$.
- We'll first apply this property, then prove it.

Q1-2

Public and Private Keys

- The first (bijection) property tells us that $x \rightarrow x^e \bmod N$ is a reasonable way to encode messages, since no information is lost
  - If you publish (N, e) as your *public key*, anyone can encrypt and send messages to you
- The second tells how to decrypt a message
  - Keep your *private key*, d, so that when you receive a message m', you can decode it by calculating $(m')^d \bmod N$.

Example (from Wikipedia)

- p=61, q=53. Compute N = pq = 3233
- (p-1)(q-1) = 60·52 = 3120
- Choose e=17 (relatively prime to 3120)
- Compute multiplicative inverse of 17 (mod 3120)
  - d = 2753 (evidence: 17·2753 = 46801 = 1 + 15·3120)
- To encrypt m=123, take $123^{17} \pmod{3233} = 855$
- To decrypt 855, take $855^{2753} \pmod{3233} = 123$
- In practice, we would use much larger numbers for p and q.

Q3-4

Recap: RSA Public-key Cryptography

- Consider a message to be a number modulo N, an n-bit number (longer messages can be broken up into n-bit pieces)
- Pick any two large primes, p and q, and let N = pq.
- **Property**: If e is any number that is relatively prime to (p-1)(q-1), then
  - the mapping $x \rightarrow x^e \bmod N$ is a bijection on $\{0, 1, \ldots, N-1\}$
  - If d is the inverse of e mod (p-1)(q-1), then for all x in $\{0, 1, \ldots, N-1\}$, $(x^e)^d \equiv x \pmod N$.
- We have applied the property, now we prove it

Proof of the property

- **Property**: If N=pq for 2 primes p and q, and if e is any number that is relatively prime to N' = (p-1)(q-1), then
  - the mapping $x \rightarrow x^e \bmod N$ is a bijection on $\{0, 1, \ldots, N-1\}$
  - If d is the inverse of e mod (p-1)(q-1), then for all x in $\{0, 1, \ldots, N-1\}$, $(x^e)^d \equiv x \pmod N$
- The 2nd condition implies the 1st, so we prove the 2nd
- e is invertible mod (p-1)(q-1) because it is relatively prime to it. Let d be its inverse
- $ed \equiv 1 \bmod (p-1)(q-1)$, so $ed = 1 + k(p-1)(q-1)$ for some integer k
- $x^{ed} - x = x^{1 + k(p-1)(q-1)} - x$. Show that this is $\equiv 0 \pmod N$
- By Fermat's Little Theorem, this expression is divisible by p. Similarly, divisible by q
- Since p and q are primes, $x^{ed} - x$ is divisible by pq = N

Q5

RSA security

- **Assumption** (Factoring is hard!):
  - Given N, e, and x^e mod N, it is computationally intractable to determine x
  - What would it take to determine x?
- Presumably this will always be true if we choose N large enough
- But people have found other ways to attack RSA, by gathering additional information
- So these days, more sophisticated techniques are needed.
- We have a MA/CSSE course in cryptography

Amortized efficiency analysis

- P49-50 in the textbook
- We analyze not just a single operation, but a sequence of operations performed on the same structure
  - We conclude something about the worst-case of the average of all of the operations in the sequence
- Example: Growable array exercise from 220/230, which we will quickly review today

Q6

Growable Array (implement ArrayList)

- An ArrayList has a *size* and a *capacity*
- The capacity is the length of the fixed-size array currently allocated to hold the list elements
- For definiteness, we start with *size*=0 and *capacity*=12
- We add a total of N items (N is not known in advance), one a t a time, each to the end of the structure
- When there is no room in the array (i.e. capacity=size and we need to add another element)
  - Allocate a new, larger array
  - copy the *size* existing elements to the new array
  - add the new element to the new array
- What is the total/average overhead (due to element copying) if
  a. we add one to the array capacity each time we have to grow it?
  b. we double the array capacity each time we have to grow it?
- Note in the second case that the amortized worst-case cost is asymptotically less than the worst case for a single element
- Every time we have to enlarge the capacity, we make it so we do not have to enlarge again soon

Q7-9

Brute Force Algorithms

- Straightforward, simple, not subtle, usually a simple application of the problem definition.
- Often not very efficient
- Easy to implement, so often the best choice if you know you'll only apply it to small input sizes