

MA/CSSE 473

Day 12



**RSA Data
Encryption**

Amortized Analysis

Brute Force

MA/CSSE 473 Day 12

- HW 5 is due now
- HW 6 due Monday
- Thursday: Guest Lecture by Bebo White
- Exam 1: Tuesday, September 30
- Blood Drive today and tomorrow. Kahn rooms.
- **Student Questions**
- RSA Cryptosystem: proof of inverse property
- Amortized analysis
- Brute force algorithms



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 $x \rightarrow x^e \pmod{N}$ is a bijection on $\{0, 1, \dots, N-1\}$
 - If d is the inverse of $e \pmod{(p-1)(q-1)}$, then for all x in $\{0, 1, \dots, N-1\}$, $(x^e)^d \equiv x \pmod{N}$.
- Yesterday we applied the property, now we prove it



Proof of the property

- **Property:** If e is any number that is relatively prime to $(p-1)(q-1)$, then
 - the mapping $x \rightarrow x^e \pmod N$ is a bijection on $\{0, 1, \dots, N-1\}$
 - If d is the inverse of $e \pmod{(p-1)(q-1)}$, then for all x in $\{0, 1, \dots, N-1\}$, $(x^e)^d \equiv x \pmod N$
- The 2nd condition implies the 1st, so we prove the 2nd
- e is invertible $\pmod{(p-1)(q-1)}$ because it is relatively prime to it. Let d be its inverse
- $ed \equiv 1 \pmod{(p-1)(q-1)}$, 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 $pq = N$



RSA security

- **Assumption** (Factoring is hard!):
 - Given N , e , and $x^e \bmod N$, 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



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), 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



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

