Private Key Cryptography

Alice (aka A) wants to send an encrypted message to Bob (aka B).

- A and B might share a private key known only to them.
- The same key serves for encryption and decryption.
- Example: Caesar's cipher $f(m) = m + 3 \bmod 26$ (shift each letter by three)
  - `WKH EXWOHU GLG LW`
  - `THE BUTLER DID IT`

This particular cryptosystem is very easy to solve

- Idea: look for common letters (E, A, T, S)

One Time Pads

Some private key systems are completely immune to cryptanalysis:

- A and B share the only two copies of a long list of random integers s_i for $i = 1, \ldots, N$.
- A sends B the message $\{m_i\}_{i=1}^{n}$ encrypted as:

$$c_i = (m_i + s_i) \bmod 26$$

- B decrypts A's message by computing $c_i - s_i \bmod 26$.

The good news: bulletproof cryptography
The bad news: horrible for e-commerce

- How do random users exchange the pad?

Public Key Cryptography

Idea of *public key cryptography* (Diffie-Hellman)

- Everyone's encryption scheme is posted publically
  - e.g. in a "telephone book"
- If A wants to send an encoded message to B, she looks up B's public key (i.e., B's encryption algorithm) in the telephone book
- But only B has the decryption key corresponding to his public key

BIG advantage: A need not know nor trust B.

There seems to be a problem though:

- If we publish the encryption key, won't everyone be able to decrypt?

Key observation: decrypting might be too hard, unless you know the key

- Computing f^{-1} could be much harder than computing f

Now the problem is to find an appropriate (f, f^{-1}) pair for which this is true

- Number theory to the rescue

RSA: Key Generation

Generating encryption/decryption keys

- Choose two very large (hundreds of digits) primes p, q.
  - This is done using probabilistic primality testing
  - Choose a random large number and check if it is prime
  - By the prime number theorem, there are lots of primes out there
- Let $n = pq$.
- Choose $e \in N$ relatively prime to $(p-1)(q-1)$. Here's how:
  - Choose e_1, e_2 prime and about $\sqrt{n}$
  - One must be relatively prime to $(p-1)(q-1)$
    - Otherwise $e_1 e_2 \mid (p-1)(q-1)$
  - Find out which one using Euclid's algorithm
- Compute d, the inverse of e modulo $(p-1)(q-1)$.
  - Can do this using using Euclidean algorithm
- Publish n and e (that's your public key)
- Keep the decryption key d to yourself.

RSA: Sending encrypted messages

How does someone send you a message?

- The message is divided into blocks each represented as a number M between 0 and n. To encrypt M, send

$$C = M^e \bmod n.$$

  - Need to use fast exponentiation ($2\log(n)$ multiplications) to do this efficiently

Example: Encrypt "stop" using $e = 13$ and $n = 2537$:

- `s t o p` $\leftrightarrow$ `18 19 14 15` $\leftrightarrow$ `1819 1415`
- $1819^{13} \bmod 2537 = 2081$ and
  $1415^{13} \bmod 2537 = 2182$ so
- `2081 2182` is the encrypted message.
- We did not need to know $p = 43, q = 59$ for that.

RSA: Decryption

If you get an encrypted message $C = M^e \bmod n$, how do you decrypt

- Compute $C^d \equiv M^{ed} \pmod{n}$.
  - Can do this quickly using fast exponentiation again

Claim: $M^{ed} \equiv M \pmod{n}$

Proof: Since $ed \equiv 1 \pmod{(p-1)(q-1)}$

- $ed \equiv 1 \pmod{p-1}$ and $ed \equiv 1 \pmod{q-1}$

Since $ed = k(p-1) + 1$ for some k,

$$M^{ed} = (M^{p-1})^k M \equiv M \pmod{p}$$

(Fermat's Little Theorem)

- True even if $p \mid M$

Similarly, $M^{ed} \equiv M \pmod{q}$

Since p, q, relatively prime, $M^{ed} \equiv M \pmod{n}$ (Theorem 10).

Note: Decryption would be easy for someone who can factor n.

- RSA depends on factoring being hard!

Digital Signatures

How can I send you a message in such a way that you're convinced it came from me (and can convince others).

- Want an analogue of a "certified" signature

Cool observation:

- To sign a message M, send $M^d \pmod{n}$
  - where (n, e) is my public key
- Recipient (and anyone else) can compute $(M^d)^e \equiv M \pmod{n}$, since M is public
- No one else could have sent this message, since no one else knows d.

Probabilistic Primality Testing

RSA requires really large primes.

- This requires testing numbers for primality.
  - Although there are now polynomial tests, the standard approach now uses probabilistic primality tests

Main idea in probabilistic primality testing algorithm:

- Choose b between 1 and n at random
- Apply an easily computable (deterministic) test $T(b, n)$ such that
  - $T(b, n)$ is true (for all b) if n is prime.
  - If n is composite, there are lots of b's for which $T(b, n)$ is false

Example: Compute $\gcd(b, n)$.

- If n is prime, $\gcd(b, n) = 1$
- If n is composite, $\gcd(b, n) \neq 1$ for some b's
  - Problem: there may not be that many witnesses

Example: Compute $b^{n-1} \bmod n$

- If n is prime $b^{n-1} \equiv 1 \pmod{n}$ (Fermat)
- Unfortunately, there are some composite numbers n such that $b^{n-1} \equiv 1 \pmod{n}$
  - These are called *Carmichael numbers*

There are tests $T(b, n)$ with the property that

- $T(b, n) = 1$ for all b if n is prime
- $T(b, n) = 0$ for at least $1/3$ of the b's if n is composite
- $T(b, n)$ is computable quickly (in polynomial time)

Constructing T requires a little more number theory

- Beyond the scope of this course.

Given such a test T, it's easy to construct a probabilistic primality test:

- Choose 100 (or 200) b's at random
- Test $T(b, n)$ for each one
- If $T(b, n) = 0$ for any b, declare b composite
  - This is definitely correct
- If $T(b, n) = 1$ for all b's you chose, declare n prime
  - This is highly likely to be correct

Security is Subtle

There are lots of ways of "misapplying" RSA, even assuming that factoring is hard.

- The public key $n = pq$, the product of two large primes
- How do you find the primes?
  - Guess a big odd number n_1, check if it's prime
  - If not, try $n_1 + 2$, then $n_1 + 4$, ...
  - Within roughly $\log(n_1)$ steps, you should find a prime;
- How do you find the second prime?
  - Guess a big odd number n_2, check if it's prime
  - ...
- Suppose, instead, you started with the first prime (call it p), and checked $p + 2$, $p + 4$, $p + 6$, ..., until you found another prime q, and used that.
  - Is that a good idea? NO!!!

If $n = pq$, then p is the first prime less than $\sqrt{n}$, and q is the first prime greater than $\sqrt{n}$.

- You can find both easily!

More to Explore

If you like number theory, consider taking

- MATH 332: Algebra and Number Theory

If you're interested in cryptography, try

- CS 487: Introduction to Cryptography

For a brief introduction to some current number theory, check out
http://math.arizona.edu/ mcleman/CoolNumbers/CoolNumbe

- The Ten Coolest Numbers
- thanks to Rob Tirrell for pointing this out

Prelim Coverage

- Chapter 0:
  - Sets
    - Operations: union, intersection, complementation, set difference
    - Proving equality of sets
  - Relations:
    - reflexive, symmetric, transitive, equivalence relations
    - transitive closure
  - Functions
    - Injective, surjective, bijective
    - Inverse function
  - Important functions and how to manipulate them:
    - exponent, logarithms, ceiling, floor, mod
  - Summation and product notation
  - Matrices (especially how to multiply them)
  - Proof and logic concepts
    - Proofs by contradiction
- Chapter 1

- You don't have to write algorithms in their notation
- You may have to *read* algorithms in their notation

- Chapter 2
  - induction vs. strong induction
  - guessing the right inductive hypothesis
  - inductive (recursive) definitions

- Number Theory - everything covered in class:
  - Fundamental Theorem of Arithmetic
  - gcd, lcm
  - Euclid's Algorithm and its extended version
  - Modular arithmetic, linear congruences
  - modular inverse and CRT
  - Fermat's little theorem
  - RSA

  You need to know all the theorems and corollaries discussed in class.

Combinatorics

Problem: How to count without counting.

- How do you figure out how many things there are with a certain property without actually enumerating all of them.

Sometimes this requires a lot of cleverness and deep mathematical insights.

But there are some standard techniques.

- That's what we'll be studying.

Sum and Product Rules

Example 1: In New Hampshire, license plates consisted of two letters followed by 3 digits. How many possible license plates are there?

Answer: 26 choices for the first letter, 26 for the second, 10 choices for the first number, the second number, and the third number:

$$26^2 \times 10^3 = 676,000$$

Example 2: A traveling salesman wants to do a tour of all 50 state capitals. How many ways can he do this?

Answer: 50 choices for the first place to visit, 49 for the second, . . . : 50! altogether.

Chapter 4 gives general techniques for solving counting problems like this. Two of the most important are:

The Sum Rule: If there are $n(A)$ ways to do A and, distinct from them, $n(B)$ ways to do B, then the number of ways to do A *or* B is $n(A) + n(B)$.

- This rule generalizes: there are $n(A) + n(B) + n(C)$ ways to do A *or* B *or* C
- In Section 4.8, we'll see what happens if the ways of doing A and B aren't distinct.

The Product Rule: If there are $n(A)$ ways to do A and $n(B)$ ways to do B, then the number of ways to do A *and* B is $n(A) \times n(B)$. This is true if the number of ways of doing A and B are independent; the number of choices for doing B is the same regardless of which choice you made for A.

- Again, this generalizes. There are $n(A) \times n(B) \times n(C)$ ways to do A *and* B *and* C

Some Subtler Examples

Example 3: If there are n Senators on a committee, in how many ways can a subcommittee be formed?

Two approaches:

1. Let N_1 be the number of subcommittees with 1 senator (n), N_2 the number of subcommittees with 2 senator ($n(n-1)/2$), ...

   According to the sum rule:

   $$N = N_1 + N_2 + \cdots + N_n$$

   - It turns out that $N_k = \frac{n!}{k!(n-k)!}$ (n choose k); this is discussed in Section 4.4
   - A subtlety: What about N_0? Do we allow subcommittees of size 0? How about size n?
     - The problem is somewhat ambiguous.

     If we allow subcommittees of size 0 and n, then there are 2^n subcommittees altogether.
     - This is the same as the number of subsets of the set of n Senators: there is a 1-1 correspondence between subsets and subcommittees.

2. Simpler method: Use the product rule!

- Each senator is either in the subcommittee or out of it: 2 possibilities for each senator:
  - $2 \times 2 \times \cdots \times 2 = 2^n$ choices altogether

General moral: In many combinatorial problems, there's more than one way to analyze the problem.

How many ways can the full committee be split into two sides on an issue?

This question is also ambiguous.

- If we care about which way each Senator voted, then the answer is again 2^n: Each subcommittee defines a split + vote (those in the subcommittee vote Yes, those out vote No); and each split + vote defines defines a subcommittee.
- If we don't care about which way each Senator voted, the answer is $2^n/2 = 2^{n-1}$.
    - This is an instance of the Division Rule.

Coping with Ambiguity

If you think a problem is ambiguous:

1. Explain why
2. Choose one way of resolving the ambiguity
3. Solve the problem according to your interpretation
    - Make sure that your interpretation doesn't render the problem totally trivial

More Examples

Example 4: How many legal configurations are there in Towers of Hanoi with n rings?

Answer: The product rule again: Each ring gets to "vote" for which pole it's on.

- Once you've decided which rings are on each pole, their order is determined.
- The total number of configurations is 3^n

Example 5: How many distinguishable ways can the letters of "computer" be arranged? How about "discrete"?

For computer, it's 8!:

- 8 choices for the first letter, for the second, ...

Is it 8! for discrete? Not quite.

- There are two e's

Suppose we called them e_1, e_2:

- There are two "versions" of each arrangement, depending on which e comes first: discre$_1$te$_2$ is the same as discre$_2$te$_1$.
- Thus, the right answer is 8!/2!

Division Rule: If there is a k-to-1 correspondence between of objects of type A with objects of type B, and there are $n(A)$ objects of type A, then there are $n(A)/k$ objects of type B.

A *k-to-1 correspondence* is an onto mapping in which every B object is the image of exactly k A objects.