Overview

Today:
- Math needed for basic public-key crypto algorithms
- RSA and Diffie-Hellman

Next:
- Read Chapter 11 (skip SHA-512 logic and SHA3 iteration function)
- Project phase 3 due in one week (March 28) - finish it!

Background / Context

Recall example "trapdoor" function from last time: Given a number $n$, how many positive integers divide evenly into $n$?
- If you know the prime factorization of $n$, this is easy.
- If you don’t know the factorization, don’t know efficient solution

How does this fit into the public key crypto model?
- Pick two large (e.g., 1024-bit) prime numbers $p$ and $q$
- Compute the product $n = p \cdot q$
- Public key is $n$ (hard to find $p$ and $q$), private is the pair $(p,q)$

Questions:
- How do we pick (or detect) large prime numbers?
- How do we use this trapdoor knowledge to encrypt?
**Prime Numbers**

A prime number is a number $p$ for which its only positive divisors are 1 and $p$.

Question: How common are prime numbers?

- The Prime Number Theorem states that there are approximately $\frac{n}{\ln n}$ prime numbers less than $n$.
- Picking a random $b$-bit number, probability that it is prime is approximately $\frac{1}{\ln(2^b)} = \frac{1}{\ln 2} \frac{1}{b} = 1.44 \frac{1}{b}$
  - For 1024-bit numbers this is about $1/710$
  - "Pick random 1024-bit numbers until one is prime" takes on average 710 trials
  - This is efficient - if we can tell when a number is prime!

**Primality Testing**

Problem: Given a number $n$, is it prime?

Basic algorithm: Try dividing all numbers 2,...,\sqrt{n} into $n$.

Question: How long does this take if $n$ is 1024 bits?

**Fermat’s Little Theorem**

To do better, we need to understand some properties of prime numbers, such as...

**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 is on page 46 of the textbook (not difficult!).
Fermat's Little Theorem - cont'd

Explore this formula for different values of $n$ and random $a$'s:

<table>
<thead>
<tr>
<th>$a$</th>
<th>$a^{n-1} \mod n$ ($n = 221$)</th>
<th>$a^{n-1} \mod n$ ($n = 331$)</th>
<th>$a^{n-1} \mod n$ ($n = 441$)</th>
<th>$a^{n-1} \mod n$ ($n = 541$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1</td>
<td>1</td>
<td>379</td>
<td>1</td>
</tr>
<tr>
<td>189</td>
<td>152</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>82</td>
<td>191</td>
<td>1</td>
<td>46</td>
<td>1</td>
</tr>
<tr>
<td>147</td>
<td>217</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>113</td>
<td>217</td>
<td>1</td>
<td>232</td>
<td>1</td>
</tr>
<tr>
<td>198</td>
<td>81</td>
<td>1</td>
<td>270</td>
<td>1</td>
</tr>
</tbody>
</table>

Question 1: What conclusion can be drawn about the primality of 221?

Question 2: What conclusion can be drawn about the primality of 331?

Primality Testing - First Attempt

Tempting (but incorrect) primality testing algorithm for $n$:

Pick random $a \in \{2, ..., n-2\}$
if $a^{n-1} \mod n \neq 1$ then return “not prime”
else return “probably prime”

Why doesn't this work?

Primality Testing - First Attempt

Tempting (but incorrect) primality testing algorithm for $n$:

Pick random $a \in \{2, ..., n-2\}$
if $a^{n-1} \mod n \neq 1$ then return “not prime”
else return “probably prime”

Why doesn't this work? **Carmichael numbers**...

Example: 2465 is obviously not prime, but

<table>
<thead>
<tr>
<th>$a$</th>
<th>$a^{n-1} \mod n$ ($n = 2465$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>189</td>
<td>1</td>
</tr>
<tr>
<td>82</td>
<td>1</td>
</tr>
<tr>
<td>147</td>
<td>1</td>
</tr>
<tr>
<td>113</td>
<td>1</td>
</tr>
<tr>
<td>198</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Not just for these a’s, but $a^{n-1} \mod n = 1$ for all a’s that are relatively prime to $n$. 
Primality Testing - Miller-Rabin

The previous idea is good, with some modifications (Note: This corrects a couple of typos in the textbook):

```
MILLER-RABIN-TEST(n)  // Assume n is odd
    Find k>0 and q odd such that n-1 = 2^k q
    Pick random a ∈ {2, ..., n-2}
    x = a^q mod n
    if x = 1 or x = n-1 then return “possible prime”
    for j = 1 to k-1 do
        x = x^2 mod n
        if x = n-1 then return “possible prime”
    return “composite”
```

If n is prime, always returns “possible prime”
If n is composite, says “possible prime” with probability < ¼

Idea: Run 50 times, and accept as prime iff all say “possible prime”

Question: What is the error probability?

Euler’s Totient Function and Theorem

Euler’s totient function: \( \phi(n) \) = number of integers from 1..n-1 that are relatively prime to n.

- If \( s(n) \) is count of 1..n-1 that share a factor with n, \( \phi(n) = n - 1 - s(n) \)
  - \( s(n) \) was our “trapdoor function” example
  - \( \phi(n) \) easy to compute if factorization of n known
  - Don’t know how to efficiently compute otherwise
- If n is product of two primes, \( n = p \cdot q \), then \( s(n) = \phi(p-1)(q-1) \)
  - So \( \phi(p \cdot q) = p^2 \cdot q - p^2 - q + 1 = p^2 + q - p - q + 1 = \phi(p) \cdot \phi(q) \)

Euler generalized Fermat’s Little Theorem to composite moduli:

Euler’s Theorem: For every a and n that are relatively prime (i.e., \( \gcd(a, n) = 1 \)),
\[ a^{\phi(n)} \equiv 1 \pmod{n} \]

Question: How does this simplify if n is prime?

RSA Algorithm

Key Generation:
- Pick two large primes p and q
- Calculate \( n = p \cdot q \) and \( \phi(n) = (p-1)(q-1) \)
- Pick a random e such that \( \gcd(e, \phi(n)) = 1 \)
- Compute \( d = e^{-1} \pmod{\phi(n)} \) [Use extended GCD algorithm]
- Public key is \( PU = (n, e) \); Private key is \( PR = (n, d) \)

Encryption of message \( M \in \{0, ..., n-1\} \):
\[ E(PU, M) = M^e \pmod{n} \]

Decryption of ciphertext \( C \in \{0, ..., n-1\} \):
\[ D(PR, C) = C^d \pmod{n} \]
RSA Algorithm

Key Generation:
- Pick two large primes $p$ and $q$
- Calculate $n=pq$ and $\phi(n)=(p-1)(q-1)$
- Pick a random $e$ such that $\gcd(e, \phi(n))=1$
- Compute $d=e^{-1} \pmod {\phi(n)}$: [Use extended GCD algorithm]
- Public key is $PU=(n,e)$: Private key is $PR=(n,d)$

Encryption of message $M \in \{0,..,n-1\}$:
$E(PU,M)=M^e \mod n$

Decryption of ciphertext $C \in \{0,..,n-1\}$:
$D(PR,C)=C^d \mod n$

Correctness - easy when $\gcd(M,n)=1$:
$D(PR,E(PU,M)) = (M^e)^d \mod n = M^{ed} \mod n = (M^{\phi(n)+1}) \mod n = (M^d)^{\phi(n)+1} \mod n = M$

RSA Example

Simple example:
- $p = 73$, $q = 89$
- $n = p\cdot q = 73\cdot 89 = 6497$
- $\phi(n) = (p-1)(q-1) = 72\cdot 88 = 6336$
- $e = 5$
- $d = 5069$ [ Note: $5\cdot 5069 = 25,345 = 4\cdot 6336 + 1$ ]

Encrypting message $M=1234$:
$1234^5 \mod 6497 = 1881$

Decrypting:
$1881^{5069} \mod 6497 = 1234$

Note: If time allows in class, more examples using Python!

The Discrete Log Problem

For every prime number $p$, there exists a primitive root (or “generator”) $g$ such that

$g^1, g^2, g^3, g^4, ..., g^{p-2}, g^{p-1}$ (all taken mod $p$)

are all distinct values (so a permutation of 1, 2, 3, ..., $p-1$).

Example: 3 is a primitive root of 17, with powers:

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3^i \mod 17$</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>13</td>
<td>5</td>
<td>15</td>
<td>11</td>
<td>16</td>
<td>14</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>12</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

$f_{g,p}(i) = g^i \mod p$ is a bijective mapping on $\{1, ..., p-1\}$

$g$ and $p$ are global public parameters

$f_{g,p}(i)$ is easy to compute (modular powering algorithm)

Inverse, written $\text{dlog}_{g,p}(x) = f_{g,p}^{-1}(x)$, is believed to be difficult to compute
**Diffie-Hellman Key Exchange**

Assume $g$ and $p$ are known, public parameters

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$ ← random value from ${1, \ldots, p-1}$</td>
<td>$b$ ← random value from ${1, \ldots, p-1}$</td>
</tr>
<tr>
<td>$A = g^a \mod p$</td>
<td>$B = g^b \mod p$</td>
</tr>
</tbody>
</table>

Send $A$ to Bob

Send $B$ to Alice

$S_a = B^a \mod p$

$S_b = A^b \mod p$

In the end, Alice’s secret ($S_a$) is the same as Bob’s secret ($S_b$):

$$S_a = B^a = g^{ab} = A^b = S_b$$

Eavesdropper knows $A$ and $B$, but to get $a$ or $b$ requires solving the discrete logarithm problem!

---

**Abstracting the Problem**

There are many sets over which we can define powering.

**Example:** Can look at powers of $n \times n$ matrices ($A^2$, $A^3$, etc.)

Any finite set $S$ with an element $g$ such that $f: S \to S$ is a bijection (where $f(x) = g^x$ for all $x \in S$) is called a **cyclic group**

- Very cool math here - see Chapter 5 for more info (optional)

If $f$ is easy to compute and $f^{-1}$ is difficult, then one can do Diffie-Hellman

“Elliptic Curves” are a mathematical object with this property

In fact: $f^{-1}$ seems to be harder to compute for Elliptic Curves than $\mathbb{Z}_p$

- Consequence: Elliptic Curves can use shorter numbers/keys than standard Diffie-Hellman - so faster and less communication required!

---

**Revisiting Key Sizes**

From NIST publication 800-57a

Issue: PK algorithms based on mathematical relationships, and can be broken with algorithms that are faster than brute force.

We spent time getting a feel for how big symmetric cipher keys needed to be

$\Rightarrow$ How big do keys in a public key system need to be?

From NIST pub 800-57a:

<table>
<thead>
<tr>
<th>Security Strength</th>
<th>Symmetric key algorithm</th>
<th>ECC (e.g., DSA-DSS)</th>
<th>EC (e.g., RSA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>2DES</td>
<td>$d = 1024$</td>
<td>$d = 1024$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r = 896$</td>
<td>$r = 1024$</td>
</tr>
<tr>
<td>112</td>
<td>3DES</td>
<td>$d = 1680$</td>
<td>$d = 2048$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r = 1152$</td>
<td>$r = 224-256$</td>
</tr>
<tr>
<td>128</td>
<td>AES-128</td>
<td>$d = 2048$</td>
<td>$d = 2048$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r = 2048$</td>
<td>$r = 224-256$</td>
</tr>
<tr>
<td>192</td>
<td>AES-192</td>
<td>$d = 3072$</td>
<td>$d = 3072$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r = 3072$</td>
<td>$r = 256-384$</td>
</tr>
<tr>
<td>256</td>
<td>AES-256</td>
<td>$d = 7680$</td>
<td>$d = 7680$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$r = 7680$</td>
<td>$r = 384-512$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$L = 15360$</td>
<td>$L = 15360$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$N = 512$</td>
<td>$N = 512$</td>
</tr>
</tbody>
</table>