Application: Public Key Cryptography

Suppose I wanted people to send me secret messages by snail mail

- Method 0.
  - I send a padlock, that only I have the key to, to everyone who might want to send me a message.
  - They send me the message in a locked box.
  - Problem 0. I need to know in advance who wants to send me a message
  - Problem 1. Any one with one of my padlocks can inspect it to discover the key.
  - Problem 2.man [sic] in the middle attacks.
- Method 1.
  - I design a key.
  - Then I design a padlock only opened by that key
  - I publish the design of the lock on my web-site
  - Inspecting the design, does not reveal the key!
  - Now anyone can send me a secret message
  - With public key cryptography, we do the mathematical equivalent

Public Key Cryptography

Can we create a way to encrypt information such that:

- anyone can encrypt a message
- only we can decrypt the message?

In one sense the answer is no

- Anyone can encrypt all possible message and see which encrypted version matches the one sent
- But, if the number of possible messages is large, this is impractical

Public key cryptography

- Encryption using publicly available information is fast
- Decryption using publicly available information is possible, but very very very slow
- There is a second, fast, method of decryption that relies on secret information

The RSA Algorithm

- I pick two different large primes p and q, each roughly 150 decimal digits long
- Let $n = p \cdot q$. Note n is about 300 decimal digits long
- I pick two integers e and d such that
$$0 < e, d < (p-1)(q-1)$$
and $ed \equiv 1 \pmod{(p-1)(q-1)}$
- Claim: If $0 \le a < n$ then $(a^e \bmod n)^d \bmod n = a$
  - $*$ To be proved later
- The numbers e and n are made public
- I keep d, p, and q secret.
- To encrypt a number a with $0 \le a < n$ compute $b = a^e \bmod n$. Transmit b to me.
- To decrypt b, I compute $b^d \bmod n$. This will equal a.
- To send a sequence of bits: Each segment of $\lfloor \log_2 n \rfloor$ bits encodes a number between 0 and $n-1$. So we split the sequence into segments and encrypt each segment.

Why is this secure?

- No one currently knows of a fast enough way to compute a from b, e, and n, without factoring n
- No one currently knows of a fast enough way to factor large numbers such as n

Why is it practical?

- There are plenty of primes of about 150 digits
- Finding primes of this size is not unreasonably hard
- (In practice the numbers used are probably prime with a very, very, very high probability)
- Finding a suitable d from e is reasonably fast
- All the encryption and decryption operations can be done reasonably fast

Why does it work?

Before we can prove that $(a^e \bmod n)^d \bmod n = a$, we need two theorems.

- The Chinese Remainder Theorem (CRT)
- Fermat's Little Theorem.

Chinese Remainder Theorem

Suppose we have two digital clocks displaying minutes.

- One repeats every 5 minutes: $0, 1, 2, 3, 4, 0, 1, ...$
- The other repeats every 12 minutes:
$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 0, 1, ...$$
- So, assuming perfect synchronization, we see
$(0, 0), (1, 1), (2, 2), (3, 3), (4, 4), (0, 5), (1, 6), (2, 7), (3, 8), ...$
- This sequence will repeat after $5 \cdot 12$ minutes. The sequence is
$$(0 \bmod 5, 0 \bmod 12), (1 \bmod 5, 1 \bmod 12), ...$$
- Q. For what pairs of numbers m , n will we get $m \cdot n$ different pairs?
- A. When m and n have no common factor. I.e. when $\gcd(m, n) = 1$.
- If we know the two remainders $(i \bmod m, i \bmod n)$, we can figure out the number of minutes i modulo $m \cdot n$
- If $\gcd(m, n) = 1$ and $a \equiv b \pmod{m}$ and $a \equiv b \pmod{n}$ then $a \equiv b \pmod{mn}$
- This is the Chinese Remainder Theorem

Fermat's Little Theorem

Consider the sequence $a^n \bmod p$ for some prime p and $0 < a < p$ and $n = 0, 1, 2, ...$

- For example take $p = 11$ and $a = 2$ then we get
$$\begin{aligned} & 2^0 \bmod 11, 2^1 \bmod 11, 2^2 \bmod 11, ... \\ = \; & 1, 2, 4, 8, 5, 10, 9, 7, 3, 6, 1, 2, 4, ... \end{aligned}$$
We get a sequence that starts with 1 and repeats after 10 numbers
- Consider $p = 11$ & $a = 3$ and also $p = 11$ & $a = 10$,
$1, 3, 9, 5, 4, 1, 3, ...$ and $1, 10, 1, 10, ...$
We get sequences with periods 5 and 2 respectively
- In fact for any a ($0 < a < p$) the period will be a divisor of $p - 1$. [Can you prove this?]
- In all three examples, items 0, 10, 20 etc. are 1
- In general, items 0, $p - 1$, $2(p - 1)$ etc. will be 1:
$$a^{p-1} \bmod p = 1$$
- We can generalize this result to any a that p does not divide
- This is Fermat's Little Theorem

Back to RSA

We need to show $(a^e \bmod n)^d \bmod n = a$ where

- $n = pq$,
- p and q are prime
- e and d are such that $0 < e, d < (p-1)(q-1)$ and $ed \equiv 1 \pmod{(p-1)(q-1)}$

Since $(i \bmod n)(j \bmod n) \bmod n = (i \cdot j) \bmod n$ we really need to show

$$a^{ed} \equiv a \pmod{n}$$

By the CRT we need only show $a^{ed} \equiv a \pmod{p}$ and $a^{ed} \equiv a \pmod{q}$

- First we show $a^{ed} \equiv a \pmod{p}$
  - ∗ If p divides a, then p also divides a^{ed} (since $ed > 0$); thus the congruence simplifies to
    $$0 \equiv 0 \pmod{p} ,$$
    which is obviously true.

  - ∗ Now suppose p does not divide a.
    Since $ed \equiv 1 \pmod{(p-1)(q-1)}$, there must be some k such that $k(p-1)(q-1) = ed - 1$.
    Let k be such that $k(p-1)(q-1) + 1 = ed$.
    $$\begin{aligned} & a^{ed} \\ = \; & a^{k(p-1)(q-1)+1} \\ = \; & a \cdot \left(a^{k(q-1)}\right)^{p-1} \end{aligned}$$
    Since p does not divide a, it also does not divide $a^{k(q-1)}$, so we can apply Fermat's little theorem. Continuing:
    $$\begin{aligned} & a^{ed} \\ = \; & a \cdot \left(a^{k(q-1)}\right)^{p-1} \\ \equiv \; & a \cdot 1 \pmod{p} \qquad \text{by Fermat's little theorem} \\ = \; & a \end{aligned}$$
    Thus $a^{ed} \equiv a \pmod{p}$
- Similarly $a^{ed} \equiv a \pmod{q}$.

Using RSA for authentication

RSA has a nice property that many public key algorithms don't.

The encryption and decryption algorithms commute.

Thus I can "sign" a message as follows.

- Suppose I have secret key d and public key (e, n).
- Suppose my message is b. With $0 \le b < n$
- I'll compute $a = b^d \bmod n$ and send you both b and a.
- On receipt, you "encrypt" a to get $b' = a^e \bmod n$ and check that $b' = b$.
- Only someone who knows d could (feasibly) have calculated a from b, n, and e.