PUBLIC KEY ENCRYPTION

1. Choose a bit-length $k$
2. Choose two primes $p$ and $q$ which can be represented with at most $k$ bits
3. Let $n = pq$ and $q(n) = (p-1)(q-1)$
4. Find $d$ such that $0 < d < n$ and $\gcd(d, q(n)) = 1$
5. Find $e$ such that $ed \mod q(n) = 1$
6. private key = $(d, n)$ and public key = $(e, n)$
7. encrypt($m$) = $m^e \mod n$ decrypt($z$) = $z^d \mod n
Cracking RSA

1. Choose a bit-length \( k \).
2. Choose two primes \( p \) and \( q \) which can be represented with at most \( k \) bits.
3. Let \( n = pq \) and \( \phi(n) = (p-1)(q-1) \).
4. Find \( d \) such that \( 0 < d < n \) and \( gcd(d,\phi(n)) = 1 \).
5. Find \( e \) such that \( de \mod \phi(n) = 1 \).
6. Private key = \((d, n)\) and public key = \((e, n)\).

Security of RSA

- \( p \): Prime number
- \( q \): Prime number
- \( n = pq \)
- \( \phi(n) = (p-1)(q-1) \)
- \( 0 < d < n \) and \( gcd(d,\phi(n)) = 1 \)
- \( e \): \( de \mod \phi(n) = 1 \)

Private key \((d, n)\) Public key \((e, n)\)

Assuming you can’t break the encryption itself (i.e., you cannot decrypt an encrypted message without the private key)

How else might you try and figure out the encrypted message?
Security of RSA

\[ p: \text{prime number}\]
\[ q: \text{prime number}\]
\[ n = pq\]
\[ \phi(n) = (p-1)(q-1)\]
\[ d: \ 0 < d < n \text{ and } \gcd(d, \phi(n)) = 1\]
\[ e: \ \text{de mod } \phi(n) = 1\]

private key \((d, n)\)  public key \((e, n)\)

Assuming you can’t break the encryption itself (i.e. you cannot decrypt an encrypted message without the private key)

Idea 2: Try and figure out the private key!

How would you do this?

For every prime \(p\) (2, 3, 5, 7 ...):
- If \(n \mod p = 0\) then \(q = n / p\)

Why do we know that this must be \(p\) and \(q\)?
Security of RSA

- \( p \): prime number
- \( q \): prime number
- \( n = pq \)
- \( \phi(n) = (p-1)(q-1) \)
- \( d \): 0 < d < n and \( \gcd(d, \phi(n)) = 1 \)
- \( e \): \( de \mod \phi(n) = 1 \)

private key \((d, n)\)  public key \((e, n)\)

For every prime \( p \) (2, 3, 5, 7 ...):
- If \( n \mod p = 0 \) then \( q = n / p \)

Since \( p \) and \( q \) are both prime, there are no other numbers that divide them evenly, therefore no other numbers divide \( n \) evenly

Security of RSA

- \( p \): prime number
- \( q \): prime number
- \( n = pq \)
- \( \phi(n) = (p-1)(q-1) \)
- \( d \): 0 < d < n and \( \gcd(d, \phi(n)) = 1 \)
- \( e \): \( de \mod \phi(n) = 1 \)

private key \((d, n)\)  public key \((e, n)\)

For every number \( p \) (2, 3, 4, 5, 6, 7 ...):
- If \( n \mod p = 0 \) then \( q = n / p \)

How long does this take?
I.e. how many \( p \) do we need to check in the worst case assuming \( n \) has \( k \) bits?

Currently, there are no known "efficient" methods for factoring a number into its primes. This is the key to why RSA works!
Implementing RSA

1. Choose a bit-length $k$

   For generating the keys, this is the only input the algorithm has

2. Choose two primes $p$ and $q$ which can be represented with at most $k$ bits

   Ideas?

Finding primes

2. Choose two primes $p$ and $q$ which can be represented with at most $k$ bits

   Idea: pick a random number and see if it's prime

   How do we check if a number is prime?

   ```python
   isPrime(num):
       for i = 2 ... sqrt(num):
           if num % i == 0:
               return false
       return true
   ```

If the number is $k$ bits, how many numbers (worst case) might we need to examine?
Finding primes

2. Choose two primes $p$ and $q$ which can be represented with at most $k$ bits
   
   Idea: pick a random number and see if it's prime
   
   - Again, with $k$ bits we can represent numbers up to $2^k$
   - Counting up to $\sqrt{2^k} = (2^{k/2}) = 2^{k/2}$

Finding primes

Primality test for $num$:
- pick a random number $a$
- perform test$(num, a)$
  - if test fails: return false
  - if test passes: return true

Does this help us?

Finding primes

Primality test for $num$:
- pick a random number $a$
- perform test$(num, a)$
  - if test fails: return false
  - if test passes: return true

If $num$ is not prime, what is the probability (chance) that we incorrectly say $num$ is a prime?

Finding primes

0.5 (50%)

Can we do any better?
Finding primes

If \( \text{num} \) is not prime, what is the probability that we incorrectly say \( \text{num} \) is a prime?

Primality test for \( \text{num} \):
- Repeat 2 times:
  - pick a random number \( a \)
  - perform test(\( \text{num}, a \))
  - if test fails: return false
- return true

\( p(0.25) \)
- Half the time we catch it on the first test
- Of the remaining half, again, half (i.e. a quarter total) we catch it on the second test
- \( 1/4 \) we don’t catch it

Finding primes

If \( \text{num} \) is not prime, what is the probability that we incorrectly say \( \text{num} \) is a prime?

Primality test for \( \text{num} \):
- Repeat 3 times:
  - pick a random number \( a \)
  - perform test(\( \text{num}, a \))
  - if test fails: return false
- return true

\( p(1/8) \)
Finding primes

Primality test for $\text{num}$:
- Repeat $m$ times:
  - pick a random number $a$
  - perform test($\text{num}, a$)
  - if test fails: return false
- return true

If num is not prime, what is the probability that we incorrectly say num is a prime?

Fermat's little theorem: If $p$ is a prime number, then for all integers $a$:

$$a^p \equiv a \pmod{p}$$

How does this help us?
Implementing RSA

1. Choose a bit-length \( k \)
2. Choose two primes \( p \) and \( q \) which can be represented with at most \( k \) bits
3. Let \( n = pq \) and \( \phi(n) = (p-1)(q-1) \)

   How do we do this?

Greatest Common Divisor

A useful property:

If two numbers are relatively prime (i.e. \( \text{gcd}(a,b) = 1 \)),
then there exists a \( c \) such that

\[ a^c \text{ mod } b = 1 \]

Greatest Common Divisor

A more useful property:

Two numbers are relatively prime (i.e. \( \text{gcd}(a,b) = 1 \))
iff there exists a \( c \) such that \( a^c \text{ mod } b = 1 \)

What does iff mean?
Greatest Common Divisor

A more useful property:

1. If two numbers are relatively prime (i.e. \( \gcd(a,b) = 1 \)), then there exists \( c \) such that \( a \cdot c \mod b = 1 \)

2. If there exists \( c \) such that \( a \cdot c \mod b = 1 \), then the two numbers are relatively prime (i.e. \( \gcd(a,b) = 1 \))

We’re going to leverage this second part

Implementing RSA

4. Find \( d \) such that \( 0 < d < n \) and \( \gcd(d, \phi(n)) = 1 \)

5. Find \( e \) such that \( d \mod \phi(n) = 1 \)

If there exists \( c \) such that \( a \cdot c \mod b = 1 \), then the two numbers are relatively prime (i.e. \( \gcd(a,b) = 1 \))

To find \( d \) and \( e \):
- pick a random \( d \), \( 0 < d < n \)
- try and find an \( e \) such that \( d \mod \phi(n) = 1 \)
  - if none exists, try another \( d \)
  - if one exists, we’re done!

Known problem with known solutions

For the assignment, I’ve provided you with a function: \( \text{inversemod} \)

Option type

Look at \( \text{option.sml} \)

http://www.cs.pomona.edu/~dkauchak/classes/cs52/examples/option.sml

option type has two constructors:
- \( \text{NONE} \) (representing no value)
- \( \text{SOME v} \) (representing the value \( v \))
case statement

```plaintext
case _______ of
    pattern1 => value
    | pattern2 => value
    | pattern3 => value
...
```

Signing documents

If a message is encrypted with the private key how can it be decrypted?

Hint:
- \((m^e)^d = m^{ed} = m \pmod{n}\)
- encrypt\((m, (e, n)) = m^e \pmod{n}\)
- decrypt\((z, (d, n)) = z^d \pmod{n}\)

encrypt\((m, (d,n)) = m^d \pmod{n}\)

decrypt\( m^d \pmod{n} , (e, n)) = (m^e)^* \pmod{n}
    = m^{ed} \pmod{n}
    = m^d \pmod{n}
    = m \quad \text{if } n < n\)
If the message can be decrypted with the public key then the sender must have had the private key.

This is a way to digitally sign a document!

Confirmed: batman likes bananas

Share your public key with everyone

How does this happen?

Anything we have to be careful of?