Option type

Look at option.sml

- option type has two constructors:
  - NONE (representing no value)
  - SOME v (representing the value v)
Public key encryption

![Diagram of public key encryption process]

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 $\text{gcd}(d,\phi(n)) = 1$
5. Find $e$ such that $de \mod \phi(n) = 1$
6. private key = $(d,n)$ and public key = $(e,n)$
7. encrypt($m$) = $m^e \mod n$ decrypt($z$) = $z^d \mod n$

RSA public key encryption

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 $\text{gcd}(d,\phi(n)) = 1$
5. Find $e$ such that $de \mod \phi(n) = 1$
6. private key = $(d,n)$ and public key = $(e,n)$
7. encrypt($m$) = $m^e \mod n$ decrypt($z$) = $z^d \mod n$

Say I maliciously intercept an encrypted message. How could I decrypt it? (Note, you can also assume that we have the public key $(e,n)$.)

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 $\text{gcd}(d,\phi(n)) = 1$
5. Find $e$ such that $de \mod \phi(n) = 1$
6. private key = $(d,n)$ and public key = $(e,n)$
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 $\text{gcd}(d,\phi(n)) = 1$
5. Find $e$ such that $de \mod \phi(n) = 1$
6. private key = $(d,n)$ and public key = $(e,n)$
7. encrypt($m$) = $m^e \mod n$ decrypt($z$) = $z^d \mod n$

Say I maliciously intercept an encrypted message. How could I decrypt it? (Note, you can also assume that we have the public key $(e,n)$.)

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 $\text{gcd}(d,\phi(n)) = 1$
5. Find $e$ such that $de \mod \phi(n) = 1$
6. private key = $(d,n)$ and public key = $(e,n)$
7. encrypt($m$) = $m^e \mod n$ decrypt($z$) = $z^d \mod n$

Say I maliciously intercept an encrypted message. How could I decrypt it? (Note, you can also assume that we have the public key $(e,n)$.)
**Cracking RSA**

encrypt(m) = m^e mod n

**Idea 1:** undo the mod operation, i.e. mod^{-1} function

If we knew m^e and e, we could figure out m

Generally, no, if we don’t know anything about the message.

The challenge is that the mod operator maps many, many numbers to a single value.

---

**Security of RSA**

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

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

Already know e and n.

If we could figure out p and q, then we could figure out the rest (i.e. d)!

---

**Security of RSA**

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

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

Already know e and n.

If we could figure out p and q, then we could figure out the rest (i.e. d)!
### Security of RSA

<table>
<thead>
<tr>
<th>p: prime number</th>
<th>( \phi(n) = (p-1)(q-1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>q: prime number</td>
<td>( n = pq )</td>
</tr>
<tr>
<td>n = pq</td>
<td></td>
</tr>
</tbody>
</table>

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

**How would you do figure out \( p \) and \( q \)?**

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.

**Why do we know that this must be \( p \) and \( q \)?**

---

<table>
<thead>
<tr>
<th>p: prime number</th>
<th>( \phi(n) = (p-1)(q-1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>q: prime number</td>
<td>( n = pq )</td>
</tr>
<tr>
<td>n = pq</td>
<td></td>
</tr>
</tbody>
</table>

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

**How long does this take?**
- I.e., how many \( p \) do we need to check in the worst case assuming \( n \) has \( k \) bits?
For every number $p$ (2, 3, 4, 5, 6, 7 …):
- If $n \mod p = 0$ then $q = n \div p$
- $p$ is at most $k$ bits
- With $k$ bits we can represent numbers up to $2^k$.
- If we assumed that $p$ was picked randomly from these numbers, then on average we’d have to check $2^{k-1}$ numbers (half of them).
- For large $k$ (e.g. 1024) this is a very big number!

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

isPrime(num):
   for $i = 2 \ldots \sqrt{\text{num}}$:
     if $\text{num} \% i == 0$:
       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

Primality test for $\text{num}$:
- pick a random number $a$
- perform test($\text{num}, a$)
  - if test fails, $\text{num}$ is not prime
  - if test passes, 50% chance that $\text{num}$ is prime

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?

0.5 (50%)

Can we do any better?

Finding primes

Primality test for `num`:
- Repeat 2 times:
  - pick a random number `a`
  - perform `test(num, a)`
    - if test fails: return false
    - if test passes: return true
- return true

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

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
• ¼ we don’t catch it
Finding primes

Primality test for `num`:
- Repeat 3 times:
  - pick a random number `a`
  - perform `test(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?

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

`p(1/8)`

Finding primes

Primality test for `num`:
- Repeat `m` times:
  - pick a random number `a`
  - perform `test(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?

Primality test for `num`:
- Repeat `m` times:
  - pick a random number `a`
  - perform `test(num, a)`
  - if test fails: return false
- return true

`p(1/2^m)`

For example, `m = 20`: `p(1/2^{20}) = p(1/1,000,000)`
Finding primes

Primality test for \( n \):
- Repeat \( m \) times:
  - pick a random number \( a \)
  - perform test\((n, a)\)
  - if test fails return false
- return true

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?

Finding primes

Fermat's little theorem: If \( p \) is a prime number, then for all integers \( a \):
\[
a^p \equiv a \pmod{p}
\]

test\((n, a)\):
- generate a random number \( a < p \)
- check if \( a^p \mod p = a \)

Implementing RSA

1. Find \( d \) such that \( 0 < d < n \) and \( \gcd(d, \phi(n)) = 1 \)
2. Find \( e \) such that \( de \mod \phi(n) = 1 \)

How do we do these steps?
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

\[ a^c \mod b = 1 \]

Greatest Common Divisor

A more useful property:

two numbers are relatively prime (i.e. \( \text{gcd}(a,b) = 1 \))

\[ \text{iff} \] there exists a \( c \) such that \( a^c \mod b = 1 \)

What does \( \text{iff} \) mean?

Implementing RSA

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

2. Find \( e \) such that \( de \mod \phi(n) = 1 \)

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

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

We're going to leverage this second part
For the assignment, I've provided you with a function: inversemod

Known problem with known solutions

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\(( m^d \pmod{n} , (e, n)) = (m^e)^d \pmod{n} = m^{ed} \pmod{n} = m^{ed} \pmod{n} = m \pmod{n} \quad \text{(if } m < n)\)

What does this do for us?
Signing documents

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

Public key encryption

How does this happen?

Anything we have to be careful of?