ENCRYPTION TAKE 2: PRACTICAL DETAILS

David Kauchak
CS52 – Spring 2015

Admin

Assignment 6

4 more assignments:
- Assignment 7 (posted), due 11/13 5pm
- Assignment 8, due 11/20 5pm
- Assignments 9 & 10, due 12/9 11:59pm

Midterm reviews Tue & Wed 7-8pm
No office hours Thursday

Courses next spring

Public key encryption

I like bananas

encrypt message

send encrypted message

decrypt message
RSA 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 $\phi(n) = (p-1)(q-1)$
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. The private key is $(d, n)$ and the public key is $(e, n)$
7. $\text{encrypt}(m) = m^e \mod n$, $\text{decrypt}(z) = z^d \mod n$

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. The private key is $(d, n)$ and the public key is $(e, n)$
7. $\text{encrypt}(m) = m^e \mod n$, $\text{decrypt}(z) = z^d \mod n$

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

- $\text{encrypt}(m) = m^e \mod n$
- **Idea 1**: undo the mod operation, i.e. $\mod^{-1}$ function

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

Do you think this is possible?

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

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

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?

Idea 2: Try and figure out the private key!

How would you do this?

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 \((e, n)\)

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

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

For every prime \( p \) (2, 3, 5, 7 …):
- If \( n \) divides \( p \) evenly then \( q = n / p \)

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

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

Implementing RSA

1. Choose a bit-length \( k \)

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

Currently, there are no known “efficient” methods for factoring a number into its primes.

This is the key to why RSA works!
Implementing RSA

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?

isPrime(num):
    for \( i = 1 \ldots \sqrt{\text{num}} \):
        if num \% i == 0:
            return false
    return true

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

- With \( k \) bits we can represent numbers up to \( 2^k \)
- We're counting up to \( \sqrt{2^k} = 2^{k/2} \)
- Which is still \( 2^{k/2} \)
- For large \( k \) (e.g. 1024) this is a very big number!
Finding primes

Primality test for \( num \):
- pick a random number \( a \)
- perform \( \text{test}(num, a) \)
  - if test fails, \( num \) is not prime
  - if test passes, 1/2 chance that \( num \) is prime

Does this help us?

Finding primes

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

If \( num \) is not prime, what are the chances that we incorrectly say \( num \) is a prime?

Finding primes

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

If \( num \) is not prime, what are the chances that we incorrectly say \( num \) is a prime?

0.5 (50%)

Can we do any better?
Primality test for $num$:
- pick a random number $a$
- perform test$(num, a)$
  - if test fails: return false
  - if test passes: 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
- ¼ we don't catch it

Finding primes

If $num$ is not prime, what are the chances 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

If $num$ is not prime, what are the chances that we incorrectly say $num$ is a prime?
Finding primes

Primality test for \textit{num}:
- Repeat \(m\) times:
  - pick a random number \(a\)
  - perform \(\text{test}(\text{num}, a)\)
  - 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 \textit{num}:
- Repeat \(m\) times:
  - pick a random number \(a\)
  - perform \(\text{test}(\text{num}, a)\)
  - 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?

Finding primes

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

test(\text{num},a):
- generate a random number \(a < p\)
- check if \(a^p \mod p = a\)

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 \(\varphi(n) = (p-1)(q-1)\)

How do we do this?
Implementing RSA

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

5. 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. $\gcd(a, b) = 1$), then there exists a $c$ such that

$$a^c \mod b = 1$$

Greatest Common Divisor

A more useful property:

1. Two numbers are relatively prime (i.e. $\gcd(a, b) = 1$) if and only if there exists a $c$ such that $a^c \mod b = 1$

What does if and only if mean?

Greatest Common Divisor

A more useful property:

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

2. If there exists a $c$ such that $a^c \mod b = 1$, 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 \( \gcd(d, \varphi(n)) = 1 \)

5. Find \( e \) such that \( de \mod \varphi(n) = 1 \)

If there exists a \( c \) such that \( a^c \mod b = 1 \), 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 \( de \mod \varphi(n) = 1 \)
- 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: \texttt{inversemod}

\texttt{inversemod}

\begin{verbatim}
|inversed : ca62int -> ca62int -> ca62int option |
\end{verbatim}

Option type

Look at \texttt{option.sml}

\texttt{http://www.cs.pomona.edu/~dkauchak/classes/cs52/examples/option.sml}

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

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

inversemod

```plaintext
(*
* inversemod : call2int -> call2int -> call2int option
* 
* inversemod u a returns (SOME v) when 0 < v < |a| and uv = 1
* (mod n). The value of v, if it exists, is unique. inversemod u a
* returns NONE if there is no such v. 
* *)
fun inversemod u a =
```

Signing documents

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

Hint:
- \((m^e)^d = m^d = m \mod n\)
- encrypt(m, (e, n)) = m^e \mod n
- decrypt(z, (d, n)) = z^d \mod n

encrypt(m, (d,n)) = m^d \mod n

decrypt( m^d \mod n , (e, n)) = (m^e)^{d \mod n}
    = m^{ed} \mod n
    = m^{ed} \mod n
    = m \text{ (if } m < n)
Signing documents

What does this do for us?

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

I like bananas

decrypt message

send signed message
Public key encryption

Share your public key with everyone

How does this happen?

Anything we have to be careful of?

What next…

More implementation details
- characters to integers
- splitting up the numbers
- finding prime numbers
- helper functions
- option type

Key distribution

“signing” documents