Arithmetic algorithms and the RSA

Grau-AA

Complexity of dealing with large integers.

We work with large single integers.

Input size: The size of an input is the number of bits required to represent that input.

N: decimal representation of the integer
n number of bits needed in the binary representation (size of input)

If $N \in \mathbb{Z}$, size $|N| = n \sim \log N$.

If an algorithm $\mathcal{A}$ has input integers $a_1, a_2, \ldots, a_k$, $\mathcal{A}$ is polynomial time if it runs in time polynomial in $\log a_1 + \log a_2 + \ldots + \log a_k$.

Review of Modular Arithmetic

Division Theorem: For any $a \in \mathbb{Z}$ and $N \in \mathbb{Z}^+$, there are unique integers q and r such that $0 \leq r < N$ and $a = qN + r$.

$q = \lfloor a/N \rfloor$ is the quocient and $r = a \mod N$ is the remainder.

Given $a, b \in \mathbb{Z}$, $N \in \mathbb{Z}^+$,
a is congruent with b modulo N
$a \mod N = b \mod N$
$a \equiv_N b$

iff $N | (a - b)$.

N partition $\mathbb{Z}$ in N equivalence classes $[a]_N$ according to their remainder modulo N:

$$[a]_N = \{a + kN | k \in \mathbb{Z}\}$$

Notice that

$$[a]_N = [b]_N \text{ iff } a \equiv_N b$$

Hence,

$$\mathbb{Z}_N = \{[a]_N | a \in \{0, 1 \ldots, N - 1\}\} = \{0, 1 \ldots, N - 1\}$$

Here, $a \in \mathbb{Z}_N$ represents $[a]_N$

$(a+b) \equiv_N (a \mod N) + (b \mod N)$.

$(a \cdot b) \equiv_N (a \mod N) \cdot (b \mod N)$.

$(a^b) \equiv_N (a \mod N)^b$.

Notice: that if $(a^b) \equiv_N 1$ then $(a \mod N)^b \equiv_N 1$

- $a(bc) \equiv_N (ab)c$ (associativity)
- $ab \equiv_N ba$ (commutativity)
- $a(b+c) \equiv_N ab + ac$ (distributivity)

These operations can help in simplifying big calculations.
For example to compute $2^{285} \mod 31$:

$$2^{285} \equiv_{31} (2^5)^{57} \equiv_{31} 32^{57} \equiv_{31} (32 \mod 31)^{57} \equiv_{31} 1^{57} \equiv_{31} 1$$

Modular Operations

Modular multiplication
INPUT: $x, y, N \in \mathbb{N}$
OUTPUT: $(x \cdot y) \mod N$.

To implement $x \cdot y \mod N$ we must do a *non-mod* multiplication $x \times y$ and divide by N, which needs $O(n^2)$ steps where $n = \max\{|x|, |y|, |N|\}$.

Modular exponentiation

Modular exponentiation
INPUT: Two n bit integers x and N, an integer exponent y
OUTPUT: $x^y \mod N$.

Obvious way: Multiply repeatedly by ($x \mod N$),
$x \mod N \rightarrow x^2 \mod N \rightarrow \ldots \rightarrow x^y \mod N$

Modular exponentiation

Modular exponentiation
INPUT: Two n bit integers x and N, an integer exponent y
OUTPUT: $x^y \mod N$.

Obvious way: Multiply repeatedly by $(x \mod N)$,
$x \mod N \to x^2 \mod N \to \ldots \to x^y \mod N$

Total cost: $O(yn^2)$, but $y = O(2^{|y|})$. Then the cost is exponential!

Modular exponentiation

Modular exponentiation
INPUT: Two n bit integers x and N, an integer exponent y
OUTPUT: $x^y \mod N$.

Obvious way: Multiply repeatedly by $(x \mod N)$,
$x \mod N \to x^2 \mod N \to \ldots \to x^y \mod N$

Total cost: $O(yn^2)$, but $y = O(2^{|y|})$. Then the cost is exponential!

Clever way: Repeating squaring,
$x \mod N \to x^2 \mod N \to x^4 \mod N \to \ldots \to x^{2^{\lfloor \log y \rfloor}} \mod N$

Modular Exponentiation

Function $modexp(x, y, N)$

if $y = 0$ **then**

  return 1

end if

$z := modexp(x, \lfloor y/2 \rfloor, N)$

if y is even **then**

  return $z^2 \mod N$

else

  return $x \cdot z^2 \mod N$

end if

Modular Exponentiation

Function $modexp(x, y, N)$

if $y = 0$ **then**

  return 1

end if

$z := modexp(x, \lfloor y/2 \rfloor, N)$

if y is even **then**

  return $z^2 \mod N$

else

  return $x \cdot z^2 \mod N$

end if

Complexity: n recursive calls, during each call it multiplies n bit numbers (doing computation modulo N saves us here!)
Total running time $O(n^3)$.

Greatest Common Divisor

GCD
INPUT: $a, b \in \mathbb{Z}$
QUESTION: Compute gcd (a, b)

Recall that given $a, b \in \mathbb{Z}$, the gcd (a, b) is the largest integer which divides a and b.

How to compute the gcd?

Greatest Common Divisor

GCD
INPUT: $a, b \in \mathbb{Z}$
QUESTION: Compute gcd (a, b)

Recall that given $a, b \in \mathbb{Z}$, the gcd (a, b) is the largest integer which divides a and b.

How to compute the gcd?

Obvious approach: factor and multiply common factors.
Factoring
INPUT: $N \in \mathbb{N}$
OUTPUT: Prime factors of N

Related problem:
Prime N
INPUT: $N \in \mathbb{N}$
QUESTION: Decide if N is prime.

Factoring is a very difficult problem!

Greatest Common Divisor

Alternative: Use the following Theorem:

Theorem (Euclid)

For any $a, b \in \mathbb{Z}$ with $a \geq b$, $gcd\ (a, b) = gcd\ (a \mod b, b)$.

Proof.

If $c \in \mathbb{Z}$ s.t. $c|a$ and $c|b$ then $c|a - b \Rightarrow \text{gcd}(a, b) \leq \text{gcd}(a - b, b)$.
If $c \in \mathbb{Z}$ s.t. $c|a - b$ and $c|b$ then $c|a \Rightarrow \text{gcd}(a, b) \geq \text{gcd}(a - b, b)$.
Therefore $\text{gcd}(a, b) = \text{gcd}(a - b, b)$. $\square$

Euclid's algorithm.

To compute gcd (a, b):

EUCLID(a, b)
if $b = 0$ **then**
  return a
else
  EUCLID$(b, a \mod b)$
end if

Euclid's algorithm.

To compute gcd (a, b):

EUCLID(a, b)
if $b = 0$ **then**
  return a
else
  EUCLID$(b, a \mod b)$
end if

Example

EUCLID(30,21) =EUCLID(21,9)= EUCLID(9,3)= EUCLID(3,0)= 3

Correctness of Euclid's algorithm

Theorem

The algorithm EUCLID is correct. Moreover for any integers $a > b \geq 0$, the total time of EUCLID (a, b) is $O(n^3)$, where $n = \max\{|a|, |b|\}$.

Proof.

The correctness follows from the previous theorem + the fact that each time we decrease b till it is 0, and then gcd $(a, 0) = a$.
On the other hand, notice that after two consecutive recursive calls the length of both a and b decrease by at least one bit. Then the base case will be reached within $2n$ recursive calls.
And since each call involves a $O(n^2)$ division, so the total time is $O(n^3)$. $\square$

Extended Euclid

Theorem

If a and b are any integers, not both zero, then $gcd(a, b)$ is the smallest positive element of the set $\{ax + by | x, y \in \mathbb{Z}\}$ of linear combinations of a and b.

An alternative and useful characterization of gcd (a, b):

Lemma

For any integers a and b, if $d|a$ and $d|b$ and $d = ax + by$ for some integers x and y, then necessarily $d = gcd\ (a, b)$.

A small extension to Euclid's algorithm is the key to dividing in the modular world.

Extended Euclid

EXT-EUCLID(a, b)
if $b = 0$ **then**
  return $(a, 1, 0)$
else
  $(d, x', y') :=$ **EXT-EUCLID** $(b, a \mod b)$
  return $(d, y', x' - \lfloor a/b \rfloor y')$
end if

Lemma

For any positive integers a and b, EXT-EUCLID (a, b) returns (d, x, y) s.t. gcd $(a, b) = d = ax + by$. The total time of EXT-EUCLID (a, b) is $O(n^3)$, where $n = \max\{|a|, |b|\}$.

Example

EXT-EUCLID(99,78)

$(d, x_1, y_1) := \textbf{EXT-EUCLID}\ (99, 78)$

$(d, x_2, y_2) := \textbf{EXT-EUCLID}\ (78, 21)$

$(d, x_3, y_3) := \textbf{EXT-EUCLID}\ (21, 15)$

$(d, x_4, y_4) := \textbf{EXT-EUCLID}\ (15, 6)$

$(d, x_5, y_5) := \textbf{EXT-EUCLID}\ (6, 3)$

$(d, x_6, y_6) := \textbf{EXT-EUCLID}\ (3, 0)$

Example

EXT-EUCLID(99,78)

$(d, x_1, y_1) :=$ **EXT-EUCLID** $(99, 78)$
$(d, x_2, y_2) :=$ **EXT-EUCLID** $(78, 21)$
$(d, x_3, y_3) :=$ **EXT-EUCLID** $(21, 15)$
$(d, x_4, y_4) :=$ **EXT-EUCLID** $(15, 6)$
$(d, x_5, y_5) :=$ **EXT-EUCLID** $(6, 3)$
$(d, x_6, y_6) :=$ **EXT-EUCLID** $(3, 0) = (3, 1, 0)$

Example

EXT-EUCLID(99,78)

$(d, x_1, y_1) :=$ **EXT-EUCLID** $(99, 78)$

$(d, x_2, y_2) :=$ **EXT-EUCLID** $(78, 21)$

$(d, x_3, y_3) :=$ **EXT-EUCLID** $(21, 15)$

$(d, x_4, y_4) :=$ **EXT-EUCLID** $(15, 6)$

$(d, x_5, y_5) :=$ **EXT-EUCLID** $(6, 3) = (3, 0, 1)$

$(d, x_6, y_6) :=$ **EXT-EUCLID** $(3, 0) = (3, 1, 0)$

Example

EXT-EUCLID(99,78)

$(d, x_1, y_1) :=$ **EXT-EUCLID** $(99, 78) = (3, -11, 14)$
$(d, x_2, y_2) :=$ **EXT-EUCLID** $(78, 21) = (3, 3, -11)$
$(d, x_3, y_3) :=$ **EXT-EUCLID** $(21, 15) = (3, -2, 3)$
$(d, x_4, y_4) :=$ **EXT-EUCLID** $(15, 6) = (3, 1, -2)$
$(d, x_5, y_5) :=$ **EXT-EUCLID** $(6, 3) = (3, 0, 1)$
$(d, x_6, y_6) :=$ **EXT-EUCLID** $(3, 0) = (3, 1, 0)$

Therefore $\gcd(99, 78) = 3 = (-11 \times 99 + 78 \times 14)$.

Modular division

- In real arithmetic:
  - Every number $a \neq 0$ has an inverse $1/a$.
  - Dividing by a is the same as multiplying by $1/a$
- In modular arithmetic,
  x is the multiplicative inverse of a modulo N if $a \cdot x \equiv_N 1$
  (if it exists!)

Modular division

- In real arithmetic:
  - Every number $a \neq 0$ has an inverse $1/a$.
  - Dividing by a is the same as multiplying by $1/a$
- In modular arithmetic,
  x is the multiplicative inverse of a modulo N if $a \cdot x \equiv_N 1$
  (if it exists!)

Lemma
For any $N > 1$, if $gcd(a, N) = 1$ then the equation $a \cdot x \equiv_N 1$ has a unique solution, modulo N. Otherwise it has no solution.

$(a^{-1} \mod N)$ denotes the multiplicative inverse of a modulo N, when a and N are relatively prime.

Modular division

Modular division
INPUT: $x, y, N \in \mathbb{N}$
OUTPUT: $(x \cdot y^{-1}) \mod N$ (if it exists!)

Modular division

Modular division
INPUT: $x, y, N \in \mathbb{N}$
OUTPUT: $(x \cdot y^{-1}) \mod N$ (if it exists!)

Define, $\mathbb{Z}_N^* = \{a | a \in \mathbb{Z}_N \wedge \gcd(a, N) = 1\}$.

Example: $\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$

Notice that $(\mathbb{Z}_N^*, \cdot_N)$ is an abelian group. Therefore, $\forall a \in \mathbb{Z}_N^*, \exists a^{-1} \in \mathbb{Z}_N^*$ the multiplicative inverse such that $a \cdot a^{-1} \equiv 1 (\mod N)$

To compute the multiplicative inverse of $a \in \mathbb{Z}_N^*$: use **EXT-EUCLID**(a, N) to get $ax + Ny = 1$ or $ax \equiv 1 (\mod N)$

Therefore, $a^{-1} \mod N$ can be computed in time $O(n^3)$.

Find the multiplicative inverse of $5 \mod 11$.

$\gcd(5, 11) = 1 \Rightarrow 5$ has multiplicative inverse in $\mathbb{Z}^*_{11}$:

EXT-EUCLID $(5, 11) = (1, -2, 1) \Rightarrow 5 \cdot (-2) \equiv 1 (\mod 11)$, and -2 is the multiplicative inverse of $5 \mod 11$.

If $\gcd(a, N) > 1 \Rightarrow a$ does not have an inverse in $\mathbb{Z}^*_N$

When working in $\mathbb{Z}_N$, the only possible division is between numbers relatively prime to N.

Example

Find the multiplicative inverse of $21 \mod 91$.
Notice $91 = 13 \cdot 7$ and $21 = 3 \cdot 7$ therefore $\gcd(91, 21) = 7 \Rightarrow 21$ does't have inverse $\mod 91$.

Find the multiplicative inverse of 3 mod 32
Equivalent to solve $3x \equiv 1 \mod 32 \Rightarrow x = 11$.

Euler's Totient function

Given N denote by $\phi(N)$, the Euler Totient function or Euler's phi function, defined as

$$\phi(N) = N \prod_{p|N} (1 - \frac{1}{p})$$

where $p|N$ is set of primes $p \neq 1$ dividing N.

Euler's Totient function

Given N denote by $\phi(N)$, the Euler Totient function or Euler's phi function, defined as

$$\phi(N) = N \prod_{p|N} (1 - \frac{1}{p})$$

where $p|N$ is set of primes $p \neq 1$ dividing N.

The size of $\mathbb{Z}_N^*$ is $\phi(N)$:

$$\phi(N) = |\mathbb{Z}_N^*|$$

If N is prime $\Rightarrow \mathbb{Z}_N^* = \{1, \ldots, N-1\}$ and $\phi(N) = N - 1$.

If N is composite $\Rightarrow \phi(N) < N - 1$.

If $N = pq$ where p and q are prime then

$$\phi(N) = N(1 - \frac{1}{p})(1 - \frac{1}{q}) = pq\frac{(p-1)(q-1)}{pq} = (p-1)(q-1).$$

Examples.

$$\mathbb{Z}_{45}^* = \{1, 2, 4, 7, 8, 11, 13, 14, 16, 17, 19, 22, 23, 26, 28, 29, 31, 32, 34, 37, 38, 41, 43, 44\}$$

As $45 = 3 \times 3 \times 5$,
$\phi(45) = 45(1 - \frac{1}{3})(1 - \frac{1}{5}) = 24$.

$\phi(35)$: As $35 = 5 \times 7 \Rightarrow \phi(35) = 4 \times 6 = 24$

Primality

Is $N \in \mathbb{N}$ prime? Erathostens sieve,

PRIME
INPUT: $N \in \mathbb{N}$
QUESTION: Decide if N is prime.

```
for a = 2, 3, ..., √N do
   if a|N then
      return "composite"
   end if
end for
return "prime"
```

Primality

Is $N \in \mathbb{N}$ prime? Erathostens sieve,

PRIME
INPUT: $N \in \mathbb{N}$
QUESTION: Decide if N is prime.

```
for a = 2, 3, ..., √N do
    if a|N then
        return "composite"
    end if
end for
return "prime"
```

Too slow!

Theorem (Euler)

For any integer $N > 1$, then

$$a^{\phi(N)} \equiv 1 \pmod N$$

for all $a \in \mathbb{Z}_N^*$.

Theorem (Fermat)

If p is prime, then

$$a^{p-1} \equiv 1 \pmod p$$

for all $a \in \mathbb{Z}_p^*$.

The Fermat Test

PRIMALITY(N)
Pick a positive integer $a < N$ at random
if $a^{N-1} \equiv 1 \pmod N$ **then**
  return yes
else
  return no
end if

The Fermat Test

PRIMALITY(N)
Pick a positive integer $a < N$ at random
if $a^{N-1} \equiv 1 \pmod{N}$ **then**
  return yes {almost sure}
else
  return no {sure}
end if

The Fermat Test

PRIMALITY(N)
Pick a positive integer $a < N$ at random
if $a^{N-1} \equiv 1 \pmod N$ **then**
  return yes {almost sure}
else
  return no {sure}
end if

Time: $O(n^3)$.

Fermat's little theorem

If N is prime,then for all $a \in \mathbb{Z}_N^*$, $a^{N-1} \equiv 1 \mod N$.

Fermat only works in one direction:

If N is prime $\Rightarrow \forall a \in \mathbb{Z}_N^*,\ a^{N-1} \equiv 1 \mod N$, but

$\exists N$ composite such that $\forall a \in \mathbb{Z}_N^*,\ a^{N-1} \equiv 1 (\mod N)$:
The Carmichael numbers
Carmichael numbers are very rare (255 with value < 100000000)
561,1105, 1729, ...
For example $561 = 3 \times 11 \times 17$

Theorem

If $a^{N-1} \not\equiv 1 \mod N$ for some $a \in \mathbb{Z}_N^*$, then this also happens with at least half of the choices $a < N$.

Sketch of the proof.

Fix some value of a for which $a^{N-1} \not\equiv 1 \mod N$ ($_{\textit{good witness of composite}}$).
The key is to notice that every element $b < N$ such that $b^{N-1} \equiv_N 1$, has a twin, $a \cdot b$ ($_{\textit{that is a good witness of composite}}$):

$$(a \cdot b)^{N-1} \equiv_N a^{N-1} \cdot b^{N-1} \equiv_N a^{N-1} \not\equiv_N 1$$

All the elements $a \cdot b$, for a fixed a but for different choices of b, are distinct (if $a \cdot i \not\equiv_N a \cdot j$ then $i \not\equiv_N j$).
The one-to-one function $b \longrightarrow a \cdot b$ shows that at least as many elements fail the test as pass it ($_{\textit{i.e. good witness of composite}}$) □

In a Carmichael-free univers

If N is prime, the previous Monte-Carlo algorithm always give the correct answer, but
if N is composite it errs with probability $\leq 1/2$.

The previous algorithm has one-side error, therefore amplifying k times the algorithm, the probability of error goes down to $\leq 1/2^k$.

Reapeated-Fermat N, k
for $i = 1$ to k **do**
  $a :=$ random $(1, N-1)$
  if $a^{N-1} \not\equiv 1 \mod N$ **then**
    return non-prime {sure}
  end if
end for
return prime {almost sure}

Taking into account the Carmichael numbers

Theorem

If N is an odd prime and $e \geq 1$, then the equation $x^2 \equiv 1 \pmod{N^e}$ has only two solutions $x \equiv 1 \pmod{N^e}$ and $x \equiv -1 \pmod{N^e}$.

A number $x \in \mathbb{Z}$ is a nontrivial root of 1, modulo N, if it satisfies $x^2 \equiv_N 1$, but $x \not\equiv_N \pm 1$. Notice that $-1 \equiv_N N - 1$.

Corollary

If there exists a nontrivial square root of 1 modulo N, then N is composite.

Example: 6 is a non-trivial root of $1 \mod 35$:
as $6^2 \equiv 1 \pmod{35}$ but $6 \neq \pm 1 \pmod{35}$.

Taking into account the Carmichael numbers.

To assure that the check of primality would not be fooled by the Carmichael numbers, given N to test for primality:

- generates several random values of $a \in \mathbb{Z}_N^+$,
- $\forall a$, checks if $a^{N-1} \equiv 1 \mod N$,
- see if it discovers a x s.t. is a non-trivial square root of 1 mod N.

Example: $5^2 \not\equiv 1 \mod 21$
$6^2 \not\equiv 1 \mod 21$
$7^2 \not\equiv 1 \mod 21$
$8^2 = 64 \equiv 1 \mod 21$
Therefore, 8 is a non-trivial root of 1 mod 21, so 21 is composite.

Witness to the compositeness of N

Given an odd integer $N > 2$, and $a \in \mathbb{Z}_N^+$, we say that a is a witness to the compositeness of N, if either:

- $a^{N-1} \not\equiv 1 \mod N$
- $\exists x_i = a^m, \exists m \in \mathbb{Z}_N^+$ s.t. x_i is a non-trivial square root of 1 $\mod N$

We define a function Witness (a, N) to test if $a^{N-1} \not\equiv 1 \mod N$ or if we can find a non-trivial root of 1 mod N.

Let $N > 2$ be odd. Then $N-1$ is even. Let $N-1 = 2^t u$ with $t > 1$ and u odd:

$N - 1 = \underbrace{101\cdots1}_{u}\underbrace{00\cdots0}_{2^t}$

For input $a \in \mathbb{Z}_N^+$, to compute $a^{N-1} \mod N$:
first compute $x_0 = a^u \mod N$, and after square the result t times
$(\cdots(x_0\underbrace{)^2\cdots)^2}_{t}$.

To go from $x_0 = a^u \mod N$ to $x_t \equiv a^{N-1} \mod N$,
we made t iterations $x_i := x_{i-1}^2 \mod N$, and check if x_{i-1} is non trivial root of 1 mod N.

Witness(a, N)
Let $N - 1 = 2^t u$ where $t \geq 1$ and u is odd
$x :=$**modexp**(a, u, N) $\{x = a^u \mod N\}$
for $i = 1$ to t **do**
  $y := x^2 \mod N$
  if $(y = 1 \wedge x \neq 1 \wedge x \neq N - 1)$ **then**
    return true $\{x$ is a non-trivial root of $1 \mod N\}$
  end if
  $x := y$
end for
if $y \neq 1$ **then**
  return true $\{a^{N-1} \not\equiv 1 \mod N$ Fermat's fail $\}$
else
  return false $\{a$ is not a witness$\}$
end if

Note: If $x_i = 1$ for some $0 \leq i < t$, **Witness** might not compute the rest of the sequence. Each value $x_{i+1}, \ldots, x_t$ would be 1.

Example: Wish to test if $a = 7$ is a witness to $N = 561$

$N - 1 = 560 = \underbrace{100011}_{u}\underbrace{0000}_{2^t} \Rightarrow u = 35, t = 4$

$x_0 = 7^{35} \mod 561 = 241$
$x_1 = 241^2 \mod 561 = 298$
$x_2 = 298^2 \mod 561 = 166$
$x_3 = 166^2 \mod 561 = 67$
$x_4 = 67^2 \mod 561 = 1$
Non-trivial root of $1 \mod 561$.

If $n = \lg N$, the complexity of **witness**(a, N) is $O(n^3)$.

Miller-Rabin primality test.

Polynomial time Monte-Carlo algorithm to decide if a given $N \in \mathbb{Z}$ is prime. The input to the algorithm would be N and the number s of $a \in \mathbb{Z}$ that we will test for witness.

Miller-Rabin(N, s)
for $i := 1$ to s **do**
  $a :=$ random $(1, N-1)$
  if witness $(a, N) =$ true **then**
    return non-prime {Definitely}
  end if
end for
return prime. {Almost surely}

If N is a n-bit number, the complexity of the algorithm is $O(sn^3)$.

Correctness

Theorem

If N is an odd composite number, the number of witnesses to the compositeness of N is $\geq \frac{N-1}{2}$.

Theorem

For any odd integer $N > 2$ and $s \in \mathbb{Z}^+$ the probability that Miller-Rabin(N, s) errs is $\leq 2^{-s}$.

Proof.

If N composite, Miller-Rabin errs if misses to discover a witness in the s iterations.
If N composite, each execution of the algorithm has probability $\geq 1/2$ of discovering a witnes a.
The probability it misses in all iteractions is $< 1/2^s$. □

Generating random numbers

We need a fast algorithm for choosing random primes that are few hundred bits long.

Theorem (Lagrange's Prime Number Theorem)

Let $\pi(N)$ *be the number of primes less than or equal to* N*. Then,* $\pi(N) \sim \frac{N}{\ln N}$*, or more precisely,*

$$\lim_{N \to \infty} \frac{\pi(N)}{(N/\ln N)} = 1$$

Generating random numbers

We need a fast algorithm for choosing random primes that are few hundred bits long.

Theorem (Lagrange's Prime Number Theorem)

Let $\pi(N)$ *be the number of primes less than or equal to* N*. Then,* $\pi(N) \sim \frac{N}{\ln N}$*, or more precisely,*

$$\lim_{N \to \infty} \frac{\pi(N)}{(N/\ln N)} = 1$$

Primes are abundant!

Algorithm to generate a random n-bit prime

- Pick a random n-bit number N.
- Run a primality test on N.
- If it passes the test, output N; else repeat the process.

How fast is this algorithm?

- If N has n-bits, the number of primes between the 2^n possible numbers is $\frac{2^n}{\ln 2^n}$.
- The probability that a randomly choosen n-bit $N \in \mathbb{Z}$ is prime is $\geq 1/n$ $\left(\frac{2^n}{\ln 2^n}/2^n = \frac{1.442}{n}\right)$.
- Therefore the expected number of Primality tests to be done until to find a prime is $O(n)$.

  For example, to choose a prime of 2000 digits will require to test 2000 randomly chosen integers.

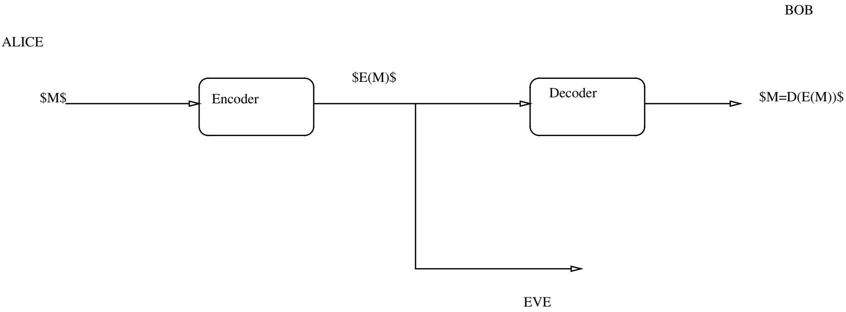
Exercise: We claim that since about a $1/n$ fraction of n-bit numbers are prime, on average it is sufficient to draw $O(n)$ random n-bit nubers before hitting a prime. Show this claim.

To generate a n-bit prime:

1. Choose a random n-number N,
2. Run Miller-Rabin on N, if passes, output N, else repeat the process.

With probability $O(1/n)$, N will be prime $\Rightarrow$ N will pass Miller-Rabin.
Otherwise, with probability $1/2^s$ Miller-Rabin errs .
To make small the failure error of Miller-Rabin take $s = \lg n$.
We need in expectation $n \lg n$ runs to get a prime.

Cryptography



A send a message M to B, E can eavesdrop M
How can we assure E can not recover M?

Private-Key Systems

Key r is secret. Both, A and B have a copy of r and $D_r = E_r^{-1}$ (dangerous)

To encrypt message M: compute $X = E(M, r) = E_r(M)$

To dencrypt X: compute
$M = D(X, r) = D(E(M, r), r) = D_r(E_r(M)) = E_r^{-1}(E_r(M))$

Public-Key Systems

(Diffie-Hellman) For each A there is a public key P_A and a secret key S_A. To know P_A does not help in discovering S_A.

A wishes to send a message to B and E is eavesdropper.
Public Keys: P_A, P_B,
Secret Keys: S_A, S_B

Secret and Public keys must have the following property:
For any person A, $M = D(E(M, P_A), S_A)$ and
$Y = E(D(Y, S_A), P_A)$

To send M from A (Alice) to B (Bob),
(1.-) A gets P_B,
(2.-) A computes the ciphertext $C = E(M, P_B)$
(3.-) A sends C to B.

When B gets C: $D(C, S_B) = D(E(M, P_B), S_B) = M$

RSA Cryptosystem: How to choose P_A and S_A

RSA : Rivest-Shamir-Adleman
Change text into numbers modulo N (ASCII)
(messages larger than N can be broken into smaller pieces).

1. Select large p and q primes
2. Compute $N = p \cdot q$
3. Compute $\phi(N) = (p-1) \cdot (q-1)$
4. Choose $c \in \mathbb{Z}^*_{\phi(N)}$
5. Compute d such that $cd \equiv 1 \mod \phi(N)$
   $d \equiv c^{-1} \mod \phi(N)$
6. $P_B = (c, N)$.
7. $S_B = (d, N)$.

RSA

- Bob chooses his public and secret Keys.
  - Bob picks two large (n-bit) random primes p and q.
  - His public key is $P_B = (c, N)$ where $N = p \cdot q$ and c is a $2n$-bit number relatively prime to $(p-1)(q-1)$.
    (A common choice is $c = 3$)
  - His secret key is $S_B = (d, N)$ where $d \equiv c^{-1} \mod \phi(N)$ can be computed using the extended Euclid algorithm.
- Alice wishes to send the message x to Bob.
  - She looks up his public key (N, c) and sends him $y = E(x, P_B) = x^c \mod N$.
  - He decodes the message by computing $D(y, S_B) = y^d \mod N = x$.

Complexity of RSA

1. Select p,q primes (Miller-Rabin)
2. $N = p \cdot q$ (The heart to security is the *difficulty to factorize N*)
3. $\phi(N) = (p-1) \cdot (q-1)$
4. Choose $c \in \mathbb{Z}^*_{\phi(N)}$ select a prime in $\mathbb{Z}_{\phi(N)}$ (or choose $c \in \{3, 5, 7, 11, \ldots\}$)
5. Compute d: $cd \equiv 1 \mod \phi(N)$ (Use EXT-EUCLID $(c, \phi(N))$ to solve $cd \equiv 1 \mod \phi(N)$)

Correctness of RSA

To see that for any $X \in \mathbb{Z}_N$, then $X = D(E(X, P_B), S_B)$ or $X = E(D(X, S_B), P_B)$.

Theorem

Let p and q be primes and let $N = pq$. For any $c \in \mathbb{Z}^*_{\phi(N)}$, $\phi(N) = (p-1)(q-1)$ and any integer $x \in \mathbb{Z}_N$ we have:

1. *The mapping $x \to x^c \mod N$ is a bijection from $\mathbb{Z}_N$ to $\mathbb{Z}_N$.*
2. *(Inverse mapping) Let $d = c^{-1} \mod \phi(N)$. Then for all $x \in \{0, \ldots, N-1\}$, $(x^c)^d \equiv x \mod N$.*

Proof of the correctness of RSA

As $c \in \mathbb{Z}^*_{\phi(N)}$, $d = c^{-1} \mod \phi(N)$ exists $(\phi(N) = (p-1)(q-1))$
Since $cd \equiv_{\phi(N)} 1$, $\exists k \in \mathbb{N}$: $cd = 1 + k\phi(N) \Rightarrow x^{cd} = x^{1+k\phi(N)}$.
By Fermat, $x^{p-1} \equiv_p 1$ and $x^{q-1} \equiv_q 1$
Then, $x^{(p-1)(q-1)} \equiv_p 1$ and $x^{(p-1)(q-1)} \equiv_q 1$
By the *Chinese Remainder Theorem*, $x^{(p-1)(q-1)} \equiv_N 1$

Hence, $x^{cd} \equiv_N (x^{1+k\phi(N)}) \equiv_N x(x^{k(p-1)(q-1)}) \equiv_N x$

$(2 \Rightarrow 1)$ Since $x \to x^c (\mod N)$ is invertible ($x^c \to x (\mod N)$)
then it must be a bijection. $\square$

The Security of RSA

Given N, c and y,
it is *computationally intractable* to determine x s.t.
$y = x^c \mod N$.

Note that:

- Eve can not experiment all the possible values of x (An exponential number of possibilities!).
- She could not try to factor N to retrieve p and q and then figure out d by inverting c modulo $(p-1)(q-1)$ (Factoring is hard!)

Example

$M = 2$

1. Select large p and q primes
2. Compute $N = p \cdot q$
3. Compute $\phi(n) = (p-1) \cdot (q-1)$
4. Choose $c \in \mathbb{Z}^*_{\phi(N)}$
5. Compute d such that $cd \equiv 1 \mod \phi(N)$
6. $P_A = (c, N)$.
7. $S_A = (d, N)$.

1. $p = 3,\ q = 17$
2. $N = 3 \times 17 = 51$
3. $\phi(51) = 2 \times 16 = 32$
4. $c = 3$
5. $d = 11$
6. $P = (3, 51)$
7. $S = (11, 51)$

To encrypt: $E(2,(3,51)) = 2^3 \mod 51 = 8$

To decrypt: $D(8,(11,51)) = 8^{11} \mod 51$

$8^2 \mod 51 = 64 \mod 51 = 13$
$8^4 \mod 51 = 169 \mod 51 = 16$
$8^5 \mod 51 = 16 \times 8 \mod 51 = 128 \mod 51 = 26$
$8^{10} \mod 51 = 26^2 \mod 51 = 13$
$8^{11} \mod 51 = 13 \times 8 \mod 51 = 2$

The hidden history

The **british GCHQ** (Government Communication Headquarter) discovered the public key scheme a few years before the **Stanford-MIT** teams, but is was considered a national secret until 1997.
So, contrary to Diffie and Hellman (Public Key, discrete logarithm,1976), Rivest, Shamir and Adleman (Public Key, factorization,1977), the mathematicians of the british GCHQ, James Ellis (1970) and Clifford Cocks (1973), remain basically unknown to almost everybody.

The Code Book by Simon Singh
Fourth State, 1999.