Algorithms Design III

Algorithms with Numbers II

Guoqiang Li
School of Software

SHANGHAI JIAO TONG UNIVERSITY

Primality

Fermat's Little Theorem

Theorem

If p is a prime, then for every $1 \leq a < p$,

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

Proof:

Let $S = \{1, 2, \ldots, p-1\}$, then multiplying these numbers by $a \pmod{p}$ is to permute them.

$a.i \pmod{p}$ are distinct for $i \in S$, and all the values are nonzero.

multiplying all numbers in each representation, then gives $(p-1)! \equiv a^{(p-1)}.(p-1)! \pmod{p}$, and thus

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

A (Problematic) Algorithm for Testing Primality

PRIMALITY(N)

Positive integer N;

Pick a positive integer $a < N$ at random;

if $a^{N-1} \equiv 1 \pmod{N}$ **then**

  return yes;

  else return no;

end

A (Problematic) Algorithm for Testing Primality

The problem is that Fermat's theorem is not an if-and-only-if condition.

- e.g. $341 = 11 \cdot 31$, and $2^{340} \equiv 1 \pmod{341}$

Our best hope: for composite N, most values of a will fail the test.

Rather than fixing an arbitrary value of a, we should choose it randomly from $\{1, \ldots, N-1\}$.

Carmichael Number

Theorem

There are composite numbers N such that for every $a < N$ relatively prime to N,

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

Example:

$$561 = 3 \cdot 11 \cdot 17$$

Non-Carmichael Number

Lemma

If $a^{N-1} \not\equiv 1 \pmod{N}$ for some a relatively prime to N, then it must hold for at least half the choices of $a < N$.

Proof:

Fix some value of a for which $a^{N-1} \not\equiv 1(\mod N)$.

Assume some $b < N$ satisfies $b^{N-1} \equiv 1(\mod N)$, then

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

For $b \neq b'$, we have

$$a \cdot b \not\equiv a \cdot b' \mod N$$

The one-to-one function $b \mapsto a \cdot b(\mod N)$ shows that at least as many elements fail the test as pass it.

Primality Testing without Carmichael Numbers

We are ignoring Carmichael numbers, so we can assert,

- If N is prime, then $a^{N-1} \equiv 1 \mod N$ for all $a < N$
- If N is not prime, then $a^{N-1} \equiv 1 \mod N$ for at most half the values of $a < N$.

Therefore, (for non-Carmichael numbers)

- Pr(PRIMALITY returns yes when N is prime)$= 1$
- Pr(PRIMALITY returns yes when N is not prime)$\leq 1/2$

Primality Testing with Low Error Probability


```
PRIMALITY2(N)
```
Positive integer N;

Pick positive integers $a_1, \ldots, a_k < N$ at random;

if $a_i^{N-1} \equiv 1 \mod N$ *for all* $1 \le i \le k$ **then**

  `return` yes;

  else `return` no;

end

- Pr(PRIMALITY2 returns yes when N is prime)$= 1$
- Pr(PRIMALITY2 returns yes when N is not prime)$\le 1/2^k$

Generating Random Primes

Lagrange's Prime Number Theorem

Let $\pi(x)$ be the number of primes $\leq x$. Then $\pi(x) \approx x/ln(x)$, or more precisely,

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

Such abundance makes it simple 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.

Generating Random Primes

Q: How fast is this algorithm?

If the randomly chosen N is truly prime, which happens with probability at least $1/n$, then it will certainly pass the test.

On each iteration, this procedure has at least a $1/n$ chance of halting.

Therefore on average it will halt within $O(n)$ rounds.

- Exercise 1.34!

Tips: Randomized Algorithm

Monte Carlo Algorithm (MC):

- Always bounded in runtime
- Correctness is random
- Examples: Primary Testing

Las Vegas Algorithm (LV):

- Always correct
- Runtime is random (small time with good probability)
- Examples: Quicksort, Hashing

Cryptography

The Typical Setting for Cryptography

Alice and Bob, who wish to communicate in private.

Eve, an eavesdropper, will go to great lengths to find out what Alice and Bob are saying.

Even Ida, an intruder, will break the rules of communications positively.

The Typical Setting for Cryptography

Alice wants to send a specific message x, written in binary, to her friend Bob.

- Alice encodes it as $e(x)$, sends it over.
- Bob applies his decryption function $d(.)$ to decode it: $d(e(x)) = x$.
- Eve, will intercept $e(x)$: for instance, she might be a sniffer on the network.
- Ida, can do anything Eve does, he may also be able to pretend to be Alice or Bob.

Ideally, $e(x)$ is chosen that without knowing $d(.)$, Eve cannot do anything with the information she has picked up.

IOW, knowing $e(x)$ tells her little or nothing about what x might be.

Private VS. Public Schemes

For centuries, cryptography was based on what we now call private-key protocols. Alice and Bob meet beforehand and choose a secret codebook.

Public-key schemes allow Alice to send Bob a message without having met him before.

Bob is able to implement a digital lock, to which only he has the key. Now by making this digital lock public, he gives Alice a way to send him a secure message.

Private-Key Schemes: One-Time Pad

SHANGHAI JIAO TONG UNIVERSITY

An encryption function:

$$e : \langle messages \rangle \to \langle encoded\ \ messages \rangle$$

e must be invertible, and is therefore a bijection.

- Alice and Bob secretly choose a binary string r of the same length as the message x that Alice will later send.
- Alice's encryption function is then a bitwise exclusive-or

$$e_r(x) = x \oplus r$$

- The function e_r is a bijection, and it is its own inverse:

$$e_r(e_r(x)) = (x \oplus r) \oplus r = x \oplus 0 = x$$

Why Secure?

Alice and Bob pick r at random.

This will ensure that if Eve intercepts the encoded message $y = e_r(x)$, she gets no information about x.

Why One-Time Pad

One-time pad is impractical and unsafe when r is repeatedly used.

Any one can get $x \oplus z$ when they know $x \oplus r$ and $z \oplus r$.

- it reveals whether the two messages begin or end the same;
- if one message contains a long sequence of zeros, then the corresponding part of the other message will be exposed.

If Ida is powerful enough that pretends to be Bob. . .

Therefore the random string that Alice and Bob share has to be the combined length of all the messages they will need to exchange.

- Random strings are costly!

AES (advanced encryption standard)

- 128-bit fixed size.
- repeatedly use
- no techniques to break are better than brute force

Public-Key Schemes

Anybody can send a message to anybody else using publicly available information, rather like addresses or phone numbers.

Each person has a public key known to the whole world and a secret key known only to himself.

When Alice wants to send message x to Bob, she encodes it using his public key.

Bob decrypts it using his secret key, to retrieve x.

Eve is welcome to see as many encrypted messages, but she will not be able to decode them, under certain assumptions.

The RSA Cryptosystem: Fundamental Property

Pick up two primes p and q and let $N = pq$.

For any e relatively prime to $(p-1)(q-1)$:

- The mapping $x \mapsto x^e \mod N$ is a bijection on $\{0, 1, \ldots N-1\}$.
- The inverse mapping is easily realized: let d be the inverse of e modulo $(p-1)(q-1)$. Then for all $x \in \{0, 1, \ldots, N-1\}$,
$$(x^e)^d \equiv x \mod N$$

The mapping $x \mapsto x^e \mod N$ is a reasonable way to encode messages x. If Bob publishes (N, e) as his public key, everyone else can use it to send him encrypted messages.

Bob retain the value d as his secret key, with which he can decode all messages that come to him by simply raising them to the d-th power modulo N.

Proof of the Property

SHANGHAI JIAO TONG UNIVERSITY

Proof:

If the mapping $x \to x^e \mod N$ is invertible, it must be a bijection; hence statement 2 implies statement 1.

To prove statement 2, observe that e is invertible modulo $(p-1)(q-1)$ because it is relatively prime to this number.

To show that $(x^e)^d \equiv x \mod N$: Since $ed \equiv 1 \mod (p-1)(q-1)$, can write $ed = 1 + k(p-1)(q-1)$ for some k.

Then

$$(x^e)^d - x = x^{ed} - x = x^{1+k(p-1)(q-1)} - x$$

$x^{1+k(p-1)(q-1)} - x$ is divisible by p (since $x^{p-1} \equiv 1 \mod p$) and likewise by q. Since p and q are primes, this expression must be divisible by $N = pq$.

RSA protocols

Bob chooses his public and secret keys:

- He starts by picking two large (n-bit) random primes p and q.
- His public key is (N, e) where $N = pq$ and e is a $2n$-bit number relatively prime to $(p-1)(q-1)$.
- his secret key is d, the inverse of e modulo $(p-1)(q-1)$.

Alice wishes to send message x to Bob

- She looks up his public key (N, e) and sends him $y = (x^e \mod N)$.
- He decodes the message by computing $y^d \mod N$.

Security Assumption of RSA

SHANGHAI JIAO TONG UNIVERSITY

The security of RSA hinges upon a simple assumption

Given N, e and $y = x^e \mod N$, it is computationally intractable to determine x.

How might Eve try to guess x

She could experiment with all possible values of x, each time checking whether $x^e \equiv y \mod N$, but this would take exponential time.

How might Eve try to guess x

she could try to factor N to retrieve p and q, and then figure out d by inverting e modulo $(p-1)(q-1)$, but we believe factoring to be hard.

Digital Signature

A digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document.

In a digital signature scheme, there are two algorithms, signing and verifying.

A signing algorithm that, given a message and a private key, produces a signature.

A signature verifying algorithm that, given a message, public key and a signature, either accepts or rejects the message's claim to authenticity.

Is Communication Safe?

Is a communication safe in the internet when cryptography is unbreakable?

- No!

The NSPK Protocol

$$
\begin{aligned}
A \longrightarrow B: &\quad \{A, N_A\}_{+K_B} \\
B \longrightarrow A: &\quad \{N_A, N_B\}_{+K_A} \\
A \longrightarrow B: &\quad \{N_B\}_{+K_B}
\end{aligned}
$$

An Attack

$$
\begin{array}{rcll}
A & \longrightarrow & I: & \{A, N_A\}_{+K_I} \\
I(A) & \longrightarrow & B: & \{A, N_A\}_{+K_B} \\
B & \longrightarrow & I(A): & \{N_A, N_B\}_{+K_A} \\
I & \longrightarrow & A: & \{N_A, N_B\}_{+K_A} \\
A & \longrightarrow & I: & \{N_B\}_{+K_I} \\
I(A) & \longrightarrow & B: & \{N_B\}_{+K_B}
\end{array}
$$

The Fixed NSPK Protocol

$$\begin{aligned} A \longrightarrow B &: \quad \{A, N_A\}_{+K_B} \\ B \longrightarrow A &: \quad \{B, N_A, N_B\}_{+K_A} \\ A \longrightarrow B &: \quad \{N_B\}_{+K_B} \end{aligned}$$

$$\begin{array}{rcll} A & \longrightarrow & I: & \{A, N_A\}_{+K_I} \\ I(A) & \longrightarrow & B: & \{A, N_A\}_{+K_B} \\ B & \longrightarrow & I(A): & \{B, N_A, N_B\}_{+K_A} \\ I & \not\longrightarrow & A: & \{I, N_A, N_B\}_{+K_A} \end{array}$$