Cryptology notes: lecture

SID-LAKHDAR Riyane

February the 15th, 2016

Contents

1 Modular arrithmetic

1.1 Prerequists: basic properties of the integers

Definition: Divisibility:
Let a and b two integers. Then a divides b (denoted as a/b) if there exists an integer c such that b = c * a.
In this case, we also say that b is a multiple of a

Property:

- $\forall c \in Z$, 1/c and c/c (reflexivity)
- If a/b and b/c then a/c (transitivity)
- If a/b and a/c then a/(b+c)
- $\forall c$ integer $c \neq 0$ a/b <=> ac/bc

Definition: prime numbers
A prime number is a positive integer $p \neq 1$ that is only divisible by (+/-) 1 and (+/-) p.
The set of prime numbers is denoted by P:
$P = \{2, 3, 5, 7, 11, ...\}$
A positive integer that is not a prime number is called a composite.

Theorem: There are infinitely many prime numbers. (see Ueclide's proof)

Proof (personal): Let suppose that the number of prime numbers is finit and let $\{\ p_1, p_2, ..., p_r\ \}$ the set of prime numbers.
Let $P = p_1 * ... * p_r + 1$. P can be:

- Prime: But $P > p_r$ by construction. Thus, it violates the initial hypothesis.
- Composite: $\exists p'$ a prime number which divides P. If $p' \in \{\ p_1, p_2, ..., p_r\ \}$, p' would divide $p_1 * ... * p_r$. Thus, to divide P it would need to also divide 1, which is impossible.
  Hence p' is a prime number $\notin \{\ p_1, p_2, ..., p_r\ \}$, which violates the initial hypothesis.

Thus we have prooved by contradiction that there are infinitely many prime numbers.

Remark:

- Let $\Pi(n)$ the number of prime numbers smaller than n. Then $\lim_{n\to\infty} \Pi(n) \approx \frac{n}{log(n)}$
- The probability that a random integer n is prime is about $\frac{1}{log(n)}$

Theorem: Fundamental theorem of arithmetic:
Every non zero integer n can be written as a product of primes:
$n = (+/-)1 * p_1^{\alpha_1} * p_2^{\alpha_2} ... * p_k^{\alpha_k}$, where $p_i \in P$, and $\alpha_i \in N$
This decomposition is unique if $p_1 < p_2 < ... < p_k$ and $\alpha > 0 \ \forall i$

Lemma: Euclid's lemma
Let p a prime number and $a, b \in Z$. Then p/ab => p/a or p/b.

Proof of the Euclid's lemma (personal): Let's write the decomposition of a and b in prime numbers: (this decomposition exists according to the fundamental theorem)

- $a = p_1^{\alpha_1} * ... * p^{\alpha} * ... * p_n^{\alpha_n}$
- $b = p_1^{\beta_1} * ... * p^{\beta} * ... * p_n^{\beta_n}$

Where the p_j are prime numbers and the α_j and $\beta_j \in \mathbb{N}$.
Thus, $a * b = p_1^{\alpha_1+\beta_1} * ... * p_i^{\alpha_i+\beta_i} * ... * p_n^{\alpha_n+\beta_n}$.
As p divides $a * b$, p must appear in the prime decomposition of $a * b$. Thus $\alpha > 0$ or $\beta > 0$ (or both). Thus p divides a or b divides b (or both).

Asymptotic notations and complexity basis:
f, g real functions, g is positive:

- f = O(g) if there exists a constant $c > 0$ such that $|f(x)| \leq c * g(x)$ for any x sufficiently large.
- f = O(g) if $\lim_{x \to \infty} \frac{f}{g}(x) = 0$
- $f \approx g$ if $\frac{f}{g}(x) goesto 1 when x goesto + infinity$
- f = +++++look at scheme 1 in my cahier++++ (g) if there exists $c_1 and c_2$ such that $c_1 * g(x) \leq |f(x)| \leq c_2 * g(x)$
- $f = \Omega(f)$ if there exists a constant c such that $f(x) \geq c * g(x)$ for x large enough

Property:

- $d = o(g)$ => f = O(g), g $\neq$ O(f)
- f equivalent to g <=> f = (1 + o(1)) g

The size of an integer a is the number of bits in the binary representation of |a|, that is $|log_2(a)| + 1$

Polynomial time algorithm: Algorithm whose running time is bounded by a polynomial in the length of the input. ie: the complexity is in $n^{O(1)}$ where n is the size of the input (exp(O(1)*log(n)))

Exponential time algorithm: Algorithm whose running time is exponential in the length of the input. ie: the complexity is in $exp^{O(1)*n}$

Sub-exponential time algorithm: Complexity is between poly and exponential. More precisely the complexity is in $L_n = exp(O(1) * n^{\alpha} * (log(n))^{1-\alpha})$ where $0 < \alpha < 1$
$\alpha = 0$ -> poly complexity
$\alpha = 1$ -> expo complexity
where n is the size of the input

1.2 Congruences

Theorem: Euclidiean division:
For a, b $\in Z$, $b \neq 0$, there exist a unique q, r $\in Z$ such that

- $a = b * q + r$
- $0 \leq r < |b|$

Definition: Congruence:
Let x, y, n $\in Z$. Then x is congruent to y modulo n if the r remainder in the division by n are the same.
In particular

$$\begin{aligned} x \equiv y[n] &<=> n|(x - y) \\ &<=> \exists k \in Z, x = k * n + y \end{aligned}$$

Property:

- This is an equivalence relation (reflexive, transitive and symmetric)
- Compatibility with addition and multiplication modulo n: $\forall a, b, a', b' \in Z$ such that $a = a' mod n$ and $b = b' mod n$. Then
  - $a + b = a' + b' mod n$
  - $a * b = a' * b' mod n$

The congruence relation partition Z into equivalent classes:
Definition: Residue classes mod m:

- Z/nZ is the set of equivalence (or residue) classes mod n for the congruence relation.
- For any integer m in a residue class, we call m a representative of that class +++++++ See scheme 2 in my cahier +++++++

Note that there are precisely n distant residue classes modulo n, given for example by 0, 1, ..., n-1

Property: (Z/nZ, +, *) is a ring

1.3 Modular exponentiation

Question: Given $x \in Z/nZ$ and e $\in N^*$. Hot to compute $x^e mod n$?

- 1st approach:
  $x^2 = x * xmodn$
  $x^3 = x^2 * xmodn$
  ...
  $x^e = x^{e-1} * xmodn$
  e multiplications, and each one is of cost $O((log_2(n))^2) => O(e*(log_2(n))^2)$.

- 2nd approach

$$e = \sum_{i=0}^{l} e_i * 2^i \text{ where } e_i \in 0, 1$$

$$x^e = x^{\sum_{i=0}^{l} e_i * 2^i}$$

$$= \prod_{i=0}^{l} (x^{2^i})^{e_i}$$

Example:

$$x^{1024}[n] = x^{2^{10}}$$

$$...$$

Property: Let $e = (e_{l-1}...e_0)_2$ the binary expression of e , ie $e = \sum_{i=0}^{l-1} e_i * 2^i$. Then

$$x^e = \prod_{i=0}^{l-1} (x^{2^i} modn)^{e_i}$$

$$= \prod_{i=0, e_i \neq 0}^{l-1} (x^{2^i}) modn$$

Algorithm: " Right to left " modulo exponentiation:

```
Input: $x \in \frac{Z}{nZ}$, $n \in N^*$, $e \in N^*$\\
Output: $y = x ^ e mod n$\\

  y = 1
  t = x mod n
  While (e != 0)
  {
      if (e = 1 mod 2)
      {
          y = y * t mod n
      }
      t = t*t mod n
      e = e >> 1
  }
```

Remark: The complexity is in $O(loge(logn)^2)$ -> Polynomial algorithm

Given $n \in N*, x \in \frac{Z}{nZ}$ and e, it easy to compute $x^e modn$.
However, there is no efficient (polynomial) algorithm which computes e given x^e, n, x ->this is called the discrete logarithm problem.

1.4 Extended Euclid algorithm

Definition: GCD, LCM, coprimality
For $a, b \in Z$, we call GCD(a, b) or $a \wedge b$ the greatest common divisor of a and b and lcm(a, b) or $a \vee b$ their least common multiple.

In particular

- x | a and x | b => x | $(a \wedge b)$
- a | m and b | m => (a V b) | m

+++++++++
Property: Gaus Lemma:
If p, q coprime and $x \in Z$ such that $p|q * x$, then $p|x$
Lemma: Bezout:
For a, b, c $\in Z$, $\exists u, v \in Z$ such that $u * a + v * b = gcd(a, b)$.

Proof:
If r is the remainder in the division of a by b: a = q*b + r $(0 \leq r < |b|)$. Then $a \wedge b = b \wedge r$.
Now let $r_0 = a$, $r_1 = b$. We compute the iteratively:

- $r_0 = r_1 * q_1 + r_2$ where $0 \leq r_i < |r_{i+1}|$
- $r_1 = r_2 * q_2 + r_3$
- ...
- $r_{n-2} = r_n - 1 * q_{n-1} + r_n$
- $r_{n-1} = r_n * q_n + r_{n+1}$ where $r_{n+1} = 0$

Thus $r_0 \wedge r_1 = r_1 \wedge r_2 = ... = r_{n-1} \wedge r_n = r_n \wedge r_{n+1} = r_n$
At the end, we have $a \wedge b$ is equal to the last non zero remainder r_n. The goal is u, v such that $a * u + b * v = a \wedge b$.
We define $(u_i) and (v_i)$ such that
$u_0 = 1 and v_0 = 0$
$u_1 = 0 and v_1 = 1$
Thus
$u_0 * a + v_1 * b = a$
$u_1 * a + v_1 * b = b$

Induction hypothesis: $u_{i-1} * a + v_{i-1} * b = r_{i-1}$
$u_i * a + v_i * b = r_i$

$$\begin{aligned}
r_{i+1} &= r_i * q_i - r_{i-1} \\
&= (u_i * a + v_i * b) * q_i - (u_{i-1} * a + v_{i-1} * b) \\
&= (u_i * q_i - u_{i-1}) * a + (v_i * q_i - v_{i-1}) * b \\
&= u_{i+1} * a + v_{i+1} * b
\end{aligned}$$

Corresponding pseudo-code:
!!!!!!!!!! To know !!!!!!!!!!

1. Euclidian algorithm:
   Input a, b integers.
   Output $a \wedge b$
   r0 = a
   r1 = b
   while ($r_0 \neq 0$) do | $tmp = r_0$ | $r_0 = r_1$ | $r_1 = tmp\% r_0$ // Remainder in div of initial $r_0 by r_i$ end loop return r_0

2. Euclidian Extended algorithm:
   Input a, b integers.
   Output $u, v such that u * a + v * b = gcd(a, b)$
   $u_0 = 1$
   $u_1 = 0$
   while b != 0 temp <- a a <- b b <- tmp % a q <- tmp % a tmp <- $u_0 - q * u_1$, u_0 <- u_1, u_1 <- tmp tmp <- $v_0 - q * v_1$, v_0 <- v_1, v_1 <- tmp return u_0, v_0

Theorem: chineese remainder thorem (CRT)
Let m, n co-prime integers. Let a and b be two integers. Then the system (S)

- $x = a mod n$
- $x = b mod m$

admits a unique solution modulo m*n

Proof:
Bezout => $\exists u, v$ such that $u * m + v * n = 1$.
Consider $x_0 = a * u * m + b * v * n$. Then

$$\begin{aligned} x_0 &= a * u * m mod n \\ &= a * 1 mod n \text{thanks to bezout} \\ &= a mod n \end{aligned}$$

If x_1 is an other solution of (S) modulo m*n, then

- $x_0 = x_1 mod n$
- $x_0 = x_1 mod m$

<=>

- $n|(x_0 - x_1)$
- $m|(x_0 - x_1)$

<=>
$m * n|(x_0 - x_1)$ <=> $x_0 = x_1 mod m * n$

1.5 Modular inverse

Definition: Let x, m > 0 be integers. We say x is invertibl mod n if there exists $y \in Z$ such that x*y = 1 mod n.
This is denoted $x^{-1} = y mod n$

Similarly, $x \in Z/nZ$ is invertible if $\exists y \in$ Z/nZ, x*y $=$ 1 mod n

Theorem:
An integer a is inversible modulo n if and only if $a \wedge n = 1$

Proof:
$<=$: $a \wedge n = 1$
$\exists u, v \in Z$ such that $a * u + b * v = 1$
$u = a^{-1} mod n$

$=>$: $\exists y \in Z/nZ$:
a*y $=$ 1 mod n
Thus, $\exists k : a * y + k * n = 1$
Thus $a \wedge n = 1$

Remark: p prime:
$a \in (z/nZ)^* = Z/nZ/0$
$=>$ a inversible
Def: Euler totient function: is defined by: $\forall n \in N^*, \Phi(n) = \#(Z/nZ)^x$: the set of invertible elements in Z/nZ.

Examples:

- $\Phi(1) = 1$
- $\Phi(2) = 1$
- $\Phi(3) = 2$
- $\Phi(4) = 2$

Property:

1. $\Phi(m * n) = \Phi(m) * \Phi(n)$ if $m \wedge n = 1$
2. $\Phi(p^e) = p^e - p^{e-1} = p^e * (1 - \frac{1}{p})$ if p is prime and e $>$ 0
3. $\Phi(n) = n * \prod_{i=1} k1 - \frac{1}{p_i}$ where $n = p_1^{\alpha_1} * p_2^{\alpha_2} * ... * p_k^{\alpha_k}$ is the factorization of n.

proof:

- (ii) p prime
  $(\mathbb{Z}/p\mathbb{Z})^* = (\mathbb{Z}/p\mathbb{Z})^* \Rightarrow \Phi(p) = p - 1$
  $e \in N^*$ $p^e \wedge x = 1$
  $0 \leq$ x $<$ p^e
  x $\wedge p^e \neq 1$
  Or x $\wedge p^e$ $=$ p
  $0 \leq k * p < p^e \rightarrow p^{e-1}$ choices for k

- (i) CRT
  $m \wedge n = 1$
  $\mathbb{Z}/n\mathbb{Z} * \mathbb{Z}/m\mathbb{Z} \xrightarrow{\sim} \mathbb{Z}/mn\mathbb{Z}$: this is a bijection
  $(\mathbb{Z}/n\mathbb{Z})^* * (\mathbb{Z}/m\mathbb{Z})^* \xrightarrow{\sim} (\mathbb{Z}/mn\mathbb{Z})^*$

  $n = p_1^{\alpha_1} * p_2^{\alpha_2} * ... * p_k^{\alpha_k}$, where p_i different prime and n > 0.

$$\begin{aligned}\Phi(n) &= \Phi(p_1^{\alpha_1}) * \Phi(p_2^{\alpha_2}) * ... * \Phi(p_k^{\alpha_k}) \\ &= p_1^{\alpha_1} * (1 - \frac{1}{p_1}) * ... * p_k^{\alpha_k} * (1 - \frac{1}{p_k}) \\ &= p_1^{\alpha_1} * ... * p_k^{\alpha_k} * \prod_{i=1}^{k} (1 - \frac{1}{p_i})\end{aligned}$$

1.6 Practical session

<u>Problem:</u>
Bank->

- 95 rallods bankndes
- 14 rallods coins

<u>Question:</u> Show that it is possible to pay any integer amount
Solution:
$95 \wedge 14 = 1$. Thus thanks to the Bezout lemma, $\exists u, v \in Z$ such that $95 * u + 14 * v = 1$ such that $95 * u + 14 * v = S$
r_i | 95 | 14 | 11 | 3 | 2 | 2 | 0
q_i | - | 6 | 1 | 3 | 1 | 2 |
u_i | 1 | 0 | 1 | -1| 4 | -5|
v_i | 0 | 1 | -6 | 7 |-27| 34|

Used formula $++++++++++++$ TODO $+++++++++++++++$
<u>Problem:</u>
a, b, c $\in Z$ such that $(a, b) \neq (0, 0)$

Show that the equation $a * x + b * y = c$ has a solution if and only if $a \wedge b | c$
<u>Solution:</u> $=>$:
If $\exists$ x, y such that $a * x + b * y = c => a \wedge b | a * x$ and $a \wedge b | b * y$
Thus $a \wedge b | c$
$<=$: If $a \wedge b | c$. Thus bezout $=> \exists u, v \in Z$ such that $a*u+b*v = +++++++$ TO DOT$++++++++++$

2 Factorization and RSA

2.1 Factorization properties

$+++++++$
down to $O(n * log(n) * log(log(n)))$ with modern algorithm for vary longer number.

But the converse factorizing integer as a product of two non trivial numbers is much harder. => observation is at the heart of RSA (rivest shamir ...)

Simplified algorithm:

- trivial division by odd numbers until $\sqrt{n}$
- Eratosten crypte
- Factorization records: $L(1/3)$

Definition of complexity $L_c(\alpha, n) =$

- $exp(c * (log(n))^c * (log(log(n)))^(1 - \alpha))$
- if $\alpha = 1$ then $L_c(1, n) = exp(c * log(n))$
- if $\alpha = 0$ then $L_c(0, n) = exp(c * log(log(n))$

This factorization is very costly. Thus in practice we use probabilistic algo which give a prime number with a propability about (90%) such as Miller-Rabin.

Simplest test of primality (using Ferma's theorem):
Ferma's theorem: if p is a prime. Then for any integer a: $\boldsymbol{a^p = a[p]}$

Thus the test for an input n is: compute $\boldsymbol{\alpha = a^n[n]}$. If $\boldsymbol{\alpha != a}$ then n is not prime. Otherwise, we can conclude nothing.

2.2 RSA and complexity

In RSA we use the multiplicative subgroup $\boldsymbol{Z/nZ}$ where $\boldsymbol{N = pq}$ is a product of two large primes.
After choosing p and q, we chose an exponentiation exponent e such that $\boldsymbol{e \wedge \phi(n) = 1}$, and compute the decryption $\boldsymbol{d = e^{-1}[\phi(n)]}$.
In particular, there exists an integer k such that $\boldsymbol{ed = 1 + k * \phi(n)}$.

Theorem Let p, q, N, e, d as above.
Then the maps

- $\boldsymbol{x \in (Z/nZ)}$ => $\boldsymbol{x^e modN}$
- $\boldsymbol{x \in (Z/nZ)}$ => $\boldsymbol{x^d modN}$

Are inverse of each others.

Proof: assume $\boldsymbol{x \wedge N = 1}$.
$\boldsymbol{x \in (Z/nZ)^x}$.
Recall the Euler -Fermat => $\boldsymbol{x^(\phi(n))}$
=> $\boldsymbol{x^{e*d} = x^{1+k*\phi(n)} = x * x^{k*\phi(n)} = x = x[n]}$ (because of Euler-Fermat th).
For the case where $\boldsymbol{x \wedge N \neq 1}$, use CRT and the fact the $\boldsymbol{p|xorq \wedge r}$.

Efficiency of RSA

- Modulus N efficient primality tests to choose randomly q and p large primes in (1024-4096 bits range).
- Computation of d is easy with extended Euclid algo knowing $\boldsymbol{\phi(n) = (p-1)*(q-1)}$
- Encryption / decryption: fast expontiation algo
- To speed up the encryption, choose a low-hamming weight exponent e. Typically, $\boldsymbol{e = 2^{16} + 1 = 65537}$
- decryption can also be sped up using Chinese Reminder Theorem
  $\boldsymbol{x = c^d[N] <=>}$
  - $\boldsymbol{x = c^{d_p} modp}$
  - $\boldsymbol{x = c^{d_q} modq}$

  with :
  - $\boldsymbol{d_p = d[p-1]}$
  - $\boldsymbol{d_q = d[q-1]}$

2.3 RSA exercise

Alice wants to send a message m = 100 to BOB with a RSA encryption.
The public key of BOB is (N, e) = (319, 11).
What do alice and bob have to compute?

2.4 RSA solution

- Alice:
  - $\boldsymbol{100^{11}[319]}$
  - $\boldsymbol{11 = 8 + 2 + 1}$
  - $\boldsymbol{100^2 = 111[319]}$
  - $\boldsymbol{100^4 = 199[319]}$
  - $\boldsymbol{100^8 = 45[319]}$
  - thus $\boldsymbol{100^{11} = 45 * 111 * 199[319]}$
  - $\boldsymbol{d = 51[319]}$
- Bob:
  - p = 11
  - q = 29
  - $\boldsymbol{N = p * q}$
  - $\boldsymbol{d = 11^{-1}[280]}$
  - Bob decrypts 265:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{265^{51}[319]} \\ &= \boldsymbol{265^{d_p}[11]} \text{where } \boldsymbol{d_p = 51 = 1[p-1]} \\ &= \boldsymbol{265^{d_p}[29]} \\ \boldsymbol{d_p = 51 = 23[28] M} &= \boldsymbol{4^{23}[29]} \end{aligned} \tag{1}$$