Introduction to cryptology (GBIN8U16)

✧

Extended GCD, RSA

Pierre Karpman
pierre.karpman@univ-grenoble-alpes.fr
https://www-ljk.imag.fr/membres/Pierre.Karpman/tea.html

2018–03–27

Greatest common divisor (GCD)

The *greatest common divisor* of two numbers a, $b \in \mathbb{N}$ is the largest number k, noted $\gcd(a,b)$ s.t. $a = km$, $b = km'$ for some m, $m' \in \mathbb{N}$

Co-primality

Two integers a, b are called *coprime* if $\gcd(a,b) = 1$

Examples:

- $\gcd(n,n) = \gcd(n,0) = n$ for any n
- $\gcd(n,1) = 1$ for any n
- $\gcd(n,kn) = n$ for any n
- $\gcd(p,q) = 1$ for any two prime numbers p, q
- $\gcd(p,n) = 1$ for any $n < p$

GCD computation

Given two integers, it is:

- Very important to be able to compute their gcd
- Very easy to do so (cool!)

$\rightsquigarrow$

A nice recurrence:

- Let $a, b \in \mathbb{N}$, $a > b$
- Then $k = \gcd(a, b) = \gcd(b, a \mod b)$
  - If $a \mod b = 0$, then $a = kb = qb \Rightarrow \gcd(a, b) = \gcd(b, 0) = b$
  - If $a \mod b = r$, then $a = km = qb + r$, $b = km'$
  - $\Rightarrow km = qkm' + r \Rightarrow k(m - qm') = r \Rightarrow$ k divides r too!

Euclid's algorithm

The previous recurrence leads to Euclid's algorithm for gcd computation

GCD computation (recursive)

Input: a, $b < a$
Output: $\gcd(a, b)$

1. If $b = 0$, return a
2. Return $\gcd(b, a \mod b)$

In practice, iterative (variant) versions may be preferable

Binary Euclid algorithm

Binary Euclid

Input: a, $b \neq 0 < a$
Output: $\gcd(a, b)$

1. Set $r \leftarrow a \mod b$, $a \leftarrow b$, $b \leftarrow r$
2. If $b = 0$, return a
3. Set $w \leftarrow 0$
4. While $a \equiv b \equiv 0 \mod 2$, set $w \leftarrow w + 1$, $a \leftarrow a/2$, $b \leftarrow b/2$
5. If a (resp. b) is even, divide it by two until it becomes odd
6. Set $t \leftarrow (a - b)/2$; If $t = 0$, return $a2^w$
7. If t is even, divide it by two until it becomes odd. Then if $t > 0$, set $a \leftarrow t$ else set $b \leftarrow -t$, then go to step 6

Binary Euclid (correctness brief)

Some quick correctness arguments

- After step 4, the contribution of 2 as a factor of $\gcd(a, b)$ is fully known as w
- Let $a' = km = 2A + 1$, $b' = km' = 2B + 1$, $k = \gcd(a', b')$, $\gcd(k, 2) = 1$
- Then $(2A + 1 - (2B + 1))/2 = A - B = k(m - m')/2 = km''$
- Then $\gcd(a', b') = \gcd((a' - b')/2, b')$ (if $(a' - b')/2 > b'$, $\gcd(b', (a' - b')/2)$ otherwise)

Why is the binary version useful?

- Divisions by two are just bit shifts!

Extended Euclid

Let a, b, $k = \gcd(a, b)$

- Then for any u, $v \in \mathbb{Z}$,
  $ua + vb = ukm + vkm' = k(um + vm') = kw$ with $w = um + vm'$
- Of particular interest are any u, v s.t. $um + vm' = 1$, then we have $ua + vb = k = \gcd(a, b)$
- One can easily compute such u, v by *extending* Euclid's algorithm

Extended Euclid (cont.)

Extended Euclid algorithm

Input: a, $b < a$

Output: $k = \gcd(a, b)$, u, v s.t. $ua + vb = k$

1. If $b = 0$, return $(k = a, u = 1, v = 0)$ $\triangleright$ $1 \times a + 0 \times 0 = a$
2. Set $r = a \mod b$, $q = a \div b$ $\triangleright$ $r = a - qb$
3. Let $(k, u', v') \leftarrow \gcd(b, r)$ $\triangleright$ $u'b + v'r = k = \gcd(a, b)$
   $\triangleright$ $u'b + v'(a - qb) = k$
   $\triangleright$ $b(u' - q) + v'a = k$
4. Return $(k, v', u' - q)$

Applications: Dividing in $\mathbb{Z}/N\mathbb{Z}$

Let a, $b \in \mathbb{Z}/N\mathbb{Z}$, one wants to compute a/b

- Assuming we know how to multiply, we just need to compute b^{-1}
- To do this, compute u, v s.t. $ub + vN = 1 = \gcd(b, N)$
  - If $\gcd(b, N) > 1$, b is not invertible $\mod N$ (why?)
- Then $ub = 1 - vN \Rightarrow ub = 1 \mod N \Rightarrow u = b^{-1}$

Exercise: use this algorithm to prove that $\mathbb{Z}/N\mathbb{Z}$ is a field iff N is prime

Digression: Little Fermat Theorem

Another possibility to find the inverse of $a \in \mathbb{Z}/N\mathbb{Z}$ when N is prime is to use the Little Fermat Theorem (LFT)

Little Fermat Theorem

Let p be a prime number, then for any $0 < a < p$, one has $a^{p-1} \equiv 1 \mod p$. This is implied by the more general formulation that for any a, $a^p \equiv a \mod p$.

Applications: Chinese Remainder Theorem

The (simple) Chinese Remainder Theorem (CRT)

Let $m_1, \ldots, m_k$ be k pairwise coprime (positive) integers ($\forall\, i, j\, \gcd(m_i, m_j) = 1$) and $x_1, \ldots, x_k$ any integers (for simplicity s.t. $0 \leq x_i < m_i$), then there is a unique $x \mod \prod_i m_i$ s.t. $x \equiv x_i \mod m_i$ for all $1 \geq i \geq k$

- Given x, m_i, it is easy to compute $x_i = x \mod m_i$
- The inverse problem is in fact also easy, using the extended Euclid algorithm

Note: This theorem is very useful! (E.g. used in the admitted Pohlig-Hellman algorithm; also nice to speed-up modular/big number arithmetic)

CRT: how?

CRT reconstruction

Input: $m_1, \ldots, m_k$, $x_1, \ldots, x_k$

Output: The unique $0 \geq x < \prod m_i$ s.t. $x \equiv x_i \mod m_i$

1. Let $M \leftarrow \prod_i m_i$
2. For all $1 \geq i \geq k$
3. $\quad M_i \leftarrow M/m_i$
4. $\quad$ Let a_i be such that $a_i M_i \equiv 1 \mod m_i$ ▷ *Computed from* $\gcd(M_i, m_i) = 1$
5. $\quad$ Let $X_i \leftarrow a_i M_i x_i$ ▷ $X_i \equiv x_i \mod m_i$; $X_i \equiv 0 \mod m_{j \neq i}$
6. Return $\sum_i X_i \mod M$

Back to Crypto: RSA

RSA (Rivest, Shamir, Adleman, 1977) in a nutshell: a family of "one-way permutations with trapdoor"

- Publicly define $\mathcal{P}$ that everyone can compute
- Knowing $\mathcal{P}$, it is "hard" to compute $\mathcal{P}^{-1}$ (even on a single point)
- There is a *trapdoor* associated w/ $\mathcal{P}$
- Knowing the trapdoor, it is easy to compute $\mathcal{P}^{-1}$ everywhere

RSA: how?

- Let p, q be two (large) prime numbers
- Let $N = pq$
- Any $0 < x < N$ s.t. $\gcd(x, N) = 1$ is invertible in $\mathbb{Z}/N\mathbb{Z}$
  - Note that knowing $x \notin (\mathbb{Z}/N\mathbb{Z})^\times \Leftrightarrow$ knowing p and q
  - Why?

Proposition: order of $(\mathbb{Z}/N\mathbb{Z})^\times$

Let N be as above, the order of the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^\times$ is equal to $(p-1)(q-1)$. (More generally, it is equal to $\varphi(N)$)

- So for any $x \in (\mathbb{Z}/N\mathbb{Z})^\times$, $x^{k\,\varphi(N)+1} = x$

RSA: more on how

- Let e be s.t. $\gcd(e, \varphi(N)) = 1$; consider $\mathcal{P} : x \mapsto x^e \mod N$
- $\mathcal{P}$ is a permutation over $(\mathbb{Z}/N\mathbb{Z})^\times$
- Knowing e, N, it is easy to compute $\mathcal{P}$
- Knowing e, $\varphi(N)$, it is easy to compute d s.t. $ed = 1 \mod \varphi(N)$
- Knowing d, x^e, it is easy to compute $x = x^{ed}$

$\Rightarrow$ We have a permutation with trapdoor, but how good is the latter?

RSA: how secure?

Knowing $ed = k\,\varphi(N) + 1$, it is easy to find $\varphi(N)$ (admitted)

Knowing $N = pq$, $\varphi(N) = (p-1)(q-1)$, it is easy to find p and q

- $\varphi(N) = pq - (p+q) + 1$; $p + q = -(\varphi(N) - N - 1)$
- For any a, b, knowing ab and $a + b$ allows to find a and b
  - Consider the polynomial $(X-a)(X-b) = X^2 - (a+b)X + ab$
  - $\Delta = (a+b)^2 - 4ab = (a-b)^2$
  - $a = ((a+b) + (a-b))/2$

$\Rightarrow$ Knowing, N, e, d, it is easy to factor N, plus:

- e does (basically) not depend on N

$\Rightarrow$ If it is easy to compute d from N, e, it is easy to factor N, and

- It is a hard problem to factor $N = pq$ when p, q are large random primes

BUT it might not be necessary to know d to (efficiently) invert $\mathcal{P}$

Tomorrow: what next?

How to (properly) use the RSA permutation family to imlement:

- Asymmetric key exchange
- Public-key signatures