Plan for today

- The extended Euclidean algorithm
- Modular arithmetic
- Fast exponentiation
- Chinese remainder theorem and the Euler ϕ-function
- RSA

Last time: $a, b \in \mathbb{N}$. gcd(a,b) can be computed in time $O(\text{size}(a) \cdot \text{size}(b))$

The extended Euclidean Algorithm

```
function exgcd(a, b)

Input: a, b ∈ ℕ0 with a ≥ b and a > 0
Output: (gcd(a, b), x, y) with

if b = 0 return   (a, 1, 0)
else
    Compute q, r ∈ ℕ with a = q · b + r, 0 ≤ r < b.
    (d, x', y') = exgcd(b, r)

    return
```

Thm: $a, b \in \mathbb{N}_0$, not both zero, then $\gcd(a,b) = \min\{x \cdot a + y \cdot b \mid x \cdot a + y \cdot b \geq 1,\ x, y \in \mathbb{Z}\}$

proof: each common divisor of a and b also divides $x \cdot a + y \cdot b$, $x, y \in \mathbb{Z}$

$\Rightarrow \gcd(a,b) \leq \min\{\ldots\}$

it remains to show: min is a common divisor of a and b.

Let $m = \min\{x \cdot a + y \cdot b : x, y \in \mathbb{Z},\ x \cdot a + y \cdot b \geq 1\}$

Suppose $m \nmid a$. $\Rightarrow a = q \cdot m + r$, $q, r \in \mathbb{Z}$, $1 \leq r < m$

then $r = a - q \cdot m = a - q(x \cdot a - y \cdot b) = \underbrace{(1 - q \cdot x)}_{\in \mathbb{Z}} \cdot a + \underbrace{q \cdot y}_{\in \mathbb{Z}} \cdot b$

↯ to m being minimal. ∎

The extended Euclidean Algorithm

Example: $7, 5$

$7 \cdot (-2) + 5 \cdot 3 = 1$

$7 = 1 \cdot 5 + 2 \quad (1, -2, 3)$

$5 = 2 \cdot 2 + 1 \quad (1, 1, -2)$

$2 = 2 \cdot 1 + 0 \quad (1, 0, 1)$

$(a, 0) \rightarrow (1, 1, 0)$

```
function exgcd(a, b)
```

Input: $a, b \in \mathbb{N}_0$ with $a \geq b$ and $a > 0$

Output: $(\gcd(a,b), x, y)$ with $x, y \in \mathbb{Z}$ and $x \cdot a + b \cdot y = \gcd(a,b)$

```
if b = 0 return (a, 1, 0)
```

$a = 1 \cdot a + 0 \cdot b$

is $\gcd(a,b)$

```
else
```

Compute $q, r \in \mathbb{N}$ with $a = q \cdot b + r, \; 0 \leq r < b$.

$(d, x', y') = \text{exgcd}(b, r)$ recall $d = x' \cdot b + y' \cdot r$.

$d = x' \cdot b + y' \cdot (a - q \cdot b)$

$= y' \cdot a + [x' - y' \cdot q] \cdot b$

```
return (d, y', x' - y' · q)
```

Analysis

Theorem

The extended Euclidean algorithm runs in time $O(size(a) \cdot \text{size}(b))$

Exercises: $|x| \leq \frac{b}{\gcd(a,b)}$, $|y| \leq \frac{a}{\gcd(a,b)}$.

Proof: Exercise.

Computing in $\mathbb{Z}_N$

Notation: $\mathbb{Z}_N \approx \mathbb{Z}/N\cdot\mathbb{Z}$

- $N \in \mathbb{N}$, $a \in \mathbb{Z}$: $[a] = \{x \in \mathbb{Z} : N \mid (a - x)\}$ $\quad [a] = \{x : x = q\cdot N + a\}$
- $\mathbb{Z}_N = (\{[a] : a \in \mathbb{Z}\}, \oplus, \odot)$ is a ring $\quad [a]_N \oplus [b]_N = [a+b]_N$, $\quad [a]\odot[b] = [a\cdot b]$.
- $\mathbb{Z}_N^*$ is (multiplicative) group of invertible elements.

$\mathbb{Z}_N^* = \{[a] : \exists [b] \text{ with } [a]\odot[b] = [1]\}$

Theorem

$a \in \mathbb{Z}_N$ is invertible if and only if $\gcd(a, N) = 1$.

Proof: "$\Leftarrow$" $\exists\, x, y \in \mathbb{Z}$ s.th. $x\cdot a + y\cdot N = 1$. Then $x\cdot a \equiv 1 \pmod N$

$\Rightarrow x \in \mathbb{Z}_N^*$

"$\Rightarrow$" $a \in \mathbb{Z}_N^*$, then $\exists x \in \mathbb{Z}_N$ s.th. $x\cdot a \equiv 1 \pmod N$, i.e. $\exists y \in \mathbb{Z}$ s.th.

$x\cdot a - 1 = y\cdot N \Leftrightarrow x\cdot a - y\cdot N = 1 \Rightarrow \gcd(a, N) = 1$ ∎

Computing the inverse

Goal: find x s.t. $a \cdot x \equiv 1 \bmod N$

- Given: $a \in \mathbb{Z}, N \in \mathbb{N}$
- Compute $x, y \in \mathbb{Z}$ with $\gcd(a, N) = x \cdot a + y \cdot N$ with extended Euclidean algorithm
- If $\gcd(a, N) \neq 1$, then $a \notin \mathbb{Z}_N^*$
- Else: $a^{-1} = x$

Running time $O(\text{sizeof}(a) \cdot \text{sizeof}(N))$

Fast exponentiation

- Given: $a, e, N \in \mathbb{N}$
- Task: Compute $a^e \pmod N$
- Suppose: e has n bits, i.e.,

$$e = [b_0, \ldots, b_{n-1}] = \sum_{j=0}^{n-1} b_i 2^i.$$

a^e, polytime in. size(a), size(e)

$a, e \in \mathbb{N}$. # of bits of a^e $\approx \log_2(a^e)$

$= e \cdot \log_2(a)$

$= 2^{\log_2(e)} \cdot \log_2(a)$

$a^e \equiv s \bmod N$, $s \in \{0, \ldots, N-1\}$.

No explosion of bit-length.

exponential in size(e).

a^e $e = \overrightarrow{110}$

$= 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2$.

$a^e = a^{1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2}$

$= (a^{2^0})^1 \cdot (a^{2^1})^1 \cdot (a^{2^2})^0$

$= (a) \quad a^2 \quad ((a^2)^2)^0$

$h=1$ $e = [1,0,1,0,0,0,1]$

$a^e = [a]^1 [a^2]^0 [(a^2)^2]^1 [a^{2^2 \cdot 2}]^0 [a^{2^{2 \cdot 2 \cdot 2}}]^0 [a^{2^{2 \cdot 2 \cdot 2 \cdot 2}}]^0 [a^{2^{2 \cdot 2 \cdot 2 \cdot 2 \cdot 2}}]^1$

h

Fast exponentiation algorithm

```
function exp(a, e, N)

Input: a, e, N ∈ ℕ
Output: h ∈ ℕ with h ≡ a^e (mod N)

h = 1, s = a

for j = 0 to n − 1
    if b_j = 1
        h = h · s (mod N)
    s = s^2 (mod N)

return h
```

e has n bits $e = [b_0 \dots b_{n-1}]$

Running time $O(\text{size}(e) \cdot M(\text{size}(N)))$

$M(n) \leq O(n^2)$ $\Rightarrow$ $O(\text{size}(e) \cdot \text{size}(N)^2)$

$M(k)$, where N has k bits and $M(\cdot)$ is time for multiplication.

reduce mod N, no explosion of bit-length.

$\leq M(k)$

Analysis

Theorem

Given $a, e, N \in \mathbb{N}$ with $0 \leq a \leq N$, one can compute $s \in \mathbb{N}$ with $s \equiv a^e \pmod{N}$ in time $O(M(\text{size}(N)) \cdot \text{size}(e))$, where $M(n)$ denotes the time required for n-bit multiplication.

(and division)

$\varphi(N)$

Definition

For $N \in \mathbb{N}$ we define $\varphi(N) = |\mathbb{Z}_N^*|$.

$\varphi(5) = |\mathbb{Z}_5^*| = 4$,

going to learn that φ is multiplicative. $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$

i.e. $|\mathbb{Z}_N^*| = \{ i : 0 \le i \le N \text{ and } \gcd(i, N) = 1 \}$

If $N_1, N_2 \in \mathbb{N}$ with $\gcd(N_1, N_2) = 1$

we have $\varphi(N_1 \cdot N_2) = \varphi(N_1) \cdot \varphi(N_2)$

$p \in \mathbb{P}$, then $\varphi(p) = p - 1$

Chinese remainder theorem

$|\mathbb{Z}_{a\cdot b}| = a\cdot b$

$|\mathbb{Z}_a \times \mathbb{Z}_b| = a\cdot b$

Theorem (is / ≅)

Suppose a and b are relatively prime integers. Then the map

$$f: \underbrace{\mathbb{Z}_{a\cdot b}}_{[x]_{a\cdot b}} \to \underbrace{\mathbb{Z}_a \times \mathbb{Z}_b}_{([x]_a, [x]_b)}$$

$$[x]_{a\cdot b} \mapsto ([x]_a, [x]_b)$$

is a ring-isomorphism.

$\mathbb{Z}_{ab} \cong \mathbb{Z}_a \times \mathbb{Z}_b$

$(a,b) + (c,d) = (a+c, b+d)$

$(a,b)\cdot(c,d) = a\cdot c, b\cdot d$

Proof: Usually one shows that f is well defined, i.e.

$[x]_{ab} = [x']_{a\cdot b}$, then $([x]_a, [x]_b) = ([x']_a, [x']_b)$

Clearly: $f([x+y]_{ab}) = f[x]_{ab} + f[y]_{ab}$ (operation + is compatible with the mapping)

$f([x\cdot y]_{ab}) = f([x]_{ab}) \cdot f([y]_{ab})$ (" · ")

Remains to show: f is surjective because of

Chinese remainder theorem

Theorem

Suppose a and b are relatively prime integers. Then the map

$$\begin{array}{rccc} f: & \mathbb{Z}_{a\cdot b} & \to & \mathbb{Z}_a \times \mathbb{Z}_b \\ & [x]_{a\cdot b} & \mapsto & ([x]_a, [x]_b) \end{array}$$

is a ring-isomorphism.

f is surjective ("onto"). Let $(m,n) \in \mathbb{Z}_a \times \mathbb{Z}_b$. To show:

$\exists\ q \in \mathbb{Z}$ s.th $f(q) = (m,n)$

We know: $\exists\ x, y \in \mathbb{Z}$ s.th

$x \cdot a + y \cdot b = 1$

$q = m \cdot y \cdot b + n \cdot x \cdot a$

$[q]_a = m \qquad [q]_b = n$

Proof

$\varphi(\cdot)$ is multiplicative

Corollary

If $a, b \in \mathbb{N}$ *and* $\gcd(a, b) = 1$, *then* $\varphi(a \cdot b) = \varphi(a) \cdot \varphi(b)$.

$\varphi(\cdot)$ and factoring

Corollary

Let $N = p_1^{e_1} \cdots p_k^{e_k}$ be the factorization of N into distinct prime numbers $p_1, \ldots, p_k$, then

$$\varphi(N) = \prod_{i=1}^{k} (p_i - 1) \cdot p_i^{e_i - 1}$$

RSA

Bob:

- Generates large (512 bits) primes p and q
- Computes $N = p \cdot q$.
- Selects *encryption exponent* e such that $\gcd(e, \phi(N)) = 1$
- Public key: (N, e)

Alice:

- Converts message to bit-string m
- Sends $s = m^e \pmod{N}$ to Bob

Bob:

- Computes $y = e^{-1} \pmod{phi(N)}$
- Computes $s^y \equiv m \pmod{N}$.

We used also Lagrange

Theorem

Let G be a finite group and H be a subgroup of G, then

$$|H| \text{ divides } |G|.$$

Corollary

Let $(G, \cdot)$ be a multiplicative group and $x \in G$, then

$$x^{|G|} = 1.$$