Counting modulo n and its time complexity

Mathematical Cryptography,
Lectures 3 - 4

Contents

Exponentiation in $\mathbb{Z}_n$

We can replace numbers by their remainders when adding or multiplying in $\mathbb{Z}_n$. Can we also somehow reduce an exponent when exponentiating in $\mathbb{Z}_n$?

Results of powers must repeat, because there are only finitely many numbers in $\mathbb{Z}_n$.

$$\text{There exist} \quad k > l \in \mathbb{N} \quad \text{so that} \quad a^k = a^l.$$

If a is invertible in $\mathbb{Z}_n$, than we get $a^{k-l} = 1$ from it. The powers of a repeat with a period $k - l$.
Is there any common period for all invertible $a \in \mathbb{Z}_n$?

Euler-Fermat's Theorem

Fermat's little theorem

For every prime p and every $a \not\equiv 0 \pmod{p}$ we have $a^{p-1} \equiv 1 \pmod{p}$.

Euler-Fermat's theorem

For every $a \in \mathbb{Z}_n$, where a is relatively prime to n, we have $a^{\varphi(n)} = 1$ in $\mathbb{Z}_n$.

It means: If a basis is relatively prime to n, we can reduce an exponent modulo $\varphi(n)$ counting in $\mathbb{Z}_n$.

Euler-Fermat's Theorem

Euler's phi function

$\varphi : \mathbb{N} \to \mathbb{N} : \varphi(n) =$ the number of integers between 0 and $(n-1)$ that are relatively prime to n

To calculate the Euler's phi function, we use these formulas:

- $\varphi(p) = p - 1$ for p prime
- $\varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p-1)$ for p prime and $e \in \mathbb{N}$
- $\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$ in case $n, m \in \mathbb{N}$ are relatively prime

Exercise

1) $\varphi(100) = \varphi(2^2 \cdot 5^2) = (4-2) \cdot (25-5) = 40$; $\varphi(1) = 1$.
If the prime factorization of n is known, we can calculate $\varphi(n)$.

2) $5^{64} = 5^4 = 13$ in $\mathbb{Z}_{18}$ because $\gcd(5, 18) = 1$ and $\varphi(18) = 6$.

Euler-Fermat's Theorem

Euler's theorem

Let $(G, \circ)$ be a finite group with n elements where 1 is the identity element. For every $a \in G$ we have $a^n = \underbrace{a \circ a \circ \ldots \circ a}_{n\text{-times}} = 1$ in G.

The Euler-Fermat's theorem is a special case of the Euler's theorem, applied to the group $(\mathbb{Z}_n^*, \cdot)$ of invertible elements in the monoid $(\mathbb{Z}_n, \cdot)$.

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n; a \text{ is relatively prime to } n\}$$

The number of elements of the group is $|\mathbb{Z}_n^*| = \varphi(n)$ and the identity element is 1.

Finding inverse elements in $\mathbb{Z}_n$

Note

The Euler-Fermat's theorem can be used to find an inverse element of a in $\mathbb{Z}_n$ too.

If a is relatively prime to n, then $a^{-1} = a^{\varphi(n)-1}$ in $\mathbb{Z}_n$.

The repeated squaring algorithm will compute this more quickly.

Note

If a is not invertible in $\mathbb{Z}_n$, then there also exist exponents $k > l$ such that $a^k = a^l$. The powers of an element a repeat with a period $k - l$, but no power equals to 1, i.e. $a^k \neq 1$ in $\mathbb{Z}_n$ for every $k > 0$. Otherwise, if $a^k = 1$ in $\mathbb{Z}_n$, then there would exist $a^{-1} = a^{k-1}$.

Repeated squaring algorithm

Repeated squaring algorithm

We compute a^b in $\mathbb{Z}_n$ by successive squaring.

We write the exponent binary: $b = (b_{k-1} \dots b_0)_2$
We create a sequence of commands $X=$"times a in $\mathbb{Z}_n$",
$S=$"square in $\mathbb{Z}_n$"as follows:
We put the S-command between every two ciphers in the binary notation, which creates k slots. We put the X-command in the slot just when 1 is in the corresponding place in the binary notation, otherwise we leave the slot empty.
We start from $a^0 = 1$ and we execute the commands from the left to the right.

Repeated squaring algorithm

Repeated squaring algorithm

Input: natural numbers a, b, n
Output: a^b in $\mathbb{Z}_n$

Let $b = (b_{k-1} \dots b_0)_2$ be the binary expansion of the exponent b.

- $c \leftarrow 1$
- for $i \leftarrow k - 1$ down to 0 do
  - $c \leftarrow c^2$ in $\mathbb{Z}_n$
  - if $b_i = 1$ then $c \leftarrow ca$ in $\mathbb{Z}_n$
- output c

Repeated squaring algorithm

Exercise

Count 2^{13} in Z_{20}.
The number $b = 13 = (1101)_2$ corresponds to the sequence of commands $XSXSSX$.
In Z_{20}: $1 \xrightarrow{X} 2 \xrightarrow{S} 4 \xrightarrow{X} 8 \xrightarrow{S} 4 \xrightarrow{S} 16 \xrightarrow{X} 12 = 2^{13}$

Note

The time complexity of the repeated squaring algorithm - it performes at most $2\log_2(b)$ multiplications in $\mathbb{Z}_n$.
The space complexity - it computes with numbers smaller than n^2.

Chinese remainder theorem

Chinese remainder theorem

Let $n_1, \ldots, n_k$ be a pairwise relatively prime family of natural numbers, and $a_1, \ldots, a_k$ be integers. Then there exists a solution to the system of congruences

$$x \equiv a_i \pmod{n_i} \quad \text{for all} \quad 1 \leq i \leq k.$$

Moreover, any two solutions are congruent modulo $n = \prod_{i=1}^{k} n_i$.

Chinese remainder theorem

Proof

A proof of an existence of a solution gives us a universal guide to solving residue systems, so we explain it here.

First we solve a special residue system for every $1 \leq i \leq k$,

$x \equiv 1 \pmod{n_i}$ and $x \equiv 0 \pmod{n_j}$ for any $j \neq i$.

The solution of the $i-$th residue system denoted as q_i could be found as follows:

$q_i = (\prod_{j \neq i} n_j) t_i$, where $t_i = (\prod_{j \neq i} n_j)^{-1}$ in $\mathbb{Z}_{n_i}$ (due to having a pairwise relatively prime family this inverse element exists)

It is easy to see that $a = \sum_{i=1}^{k} a_i q_i$ solves the given residue system.

Chinese remainder theorem

Exercise

Solve the residue system:
$x \equiv 2 \pmod 4$, $x \equiv 0 \pmod 5$, $x \equiv 1 \pmod 9$, $x \equiv 2 \pmod{11}$
We use the notation q_{n_i} instead of q_i.

$q_4 = 5 \cdot 9 \cdot 11 \cdot t$, where t is calculated in $\mathbb{Z}_4$: $t = (1 \cdot 1 \cdot 3)^{-1} = 3$.
Hence $q_4 = 1485$ in $\mathbb{Z}$. By analogy, calculate the other q_{n_i},
we get $q_4 = 1485$, $q_5 = 396$, $q_9 = 1540$, $q_{11} = 540$.

Finally $x = 2q_4 + 0q_5 + 1q_9 + 2q_{11} = 1630$ is the only solution in $\mathbb{Z}_{1980}$, where $1980 = 4 \cdot 5 \cdot 9 \cdot 11$.

Residue systems in general

In case $n_1, \ldots, n_k$ are not necessarily a pairwise relatively prime family of natural numbers, then the system of equations

$$x \equiv a_i \pmod{n_i} \quad \text{for all} \quad 1 \leq i \leq k$$

may or may not have a solution. If the system has a solution, then any two solutions are congruent modulo $n = \mathrm{lcm}(n_1, \ldots, n_k)$.

Exercise

1) The system $x \equiv 1 \pmod{2}$, $x \equiv 0 \pmod{4}$ has no solution.

2) The system $x \equiv 1 \pmod{2}$, $x \equiv 3 \pmod{4}$, $x \equiv 1 \pmod{5}$ has solutions $x = 11 + 20t$ for each $t \in \mathbb{Z}$.
We can found x by solving Diophantine equations obtained from: $x = 2k + 1 = 4l + 3 = 5m + 1$ for $k, l, m \in \mathbb{Z}$.

Residual arithmetic

Theorem

Let $n_1, \ldots, n_k$ be a pairwise relatively prime family of natural numbers and let $n = \prod_{i=1}^{k} n_i$. Define the map $\theta : \mathbb{Z}_n \to \mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_k} : [a]_n \mapsto ([a]_{n_1}, \ldots, [a]_{n_k})$

1. The definition is correct, it does not depend on a choice of the representative of the class $[a]_n$.
2. The map θ is a bijection.
3. For all $\alpha, \beta \in \mathbb{Z}_n$, $\theta(\alpha) = (\alpha_1, \ldots, \alpha_k)$, $\theta(\beta) = (\beta_1, \ldots, \beta_k)$, the following holds:
   $\theta(\alpha + \beta) = (\alpha_1 + \beta_1, \ldots, \alpha_k + \beta_k)$, $\theta(0) = (0, \ldots, 0)$,
   $\theta(-\alpha) = (-\alpha_1, \ldots, -\alpha_k)$;
   $\theta(\alpha \cdot \beta) = (\alpha_1 \cdot \beta_1, \ldots, \alpha_k \cdot \beta_k)$, $\theta(1) = (1, \ldots, 1)$,
   $\alpha \in \mathbb{Z}_n^*$ iff every $\alpha_i \in \mathbb{Z}_{n_i}^*$. Then $\theta(\alpha^{-1}) = (\alpha_1^{-1}, \ldots, \alpha_k^{-1})$.

Residual arithmetic

Notes

- The map θ could be defined over the base representatives of the classes in $\mathbb{Z}_n$, i.e. over the remainders after dividing by n, as follows:

  Let $n_1, \ldots, n_k$ be a pairwise relatively prime family of natural numbers, let $n = \prod_{i=1}^{k} n_i$.
  For any $0 \leq a < n$, denote its remainder after division by n_i as a_i, so $a \equiv a_i \pmod{n_i}$, $0 \leq a_i < n_i$. Then the map

  $$\theta : \mathbb{Z}_n \to \mathbb{Z}_{n_1} \times \ldots \mathbb{Z}_{n_k} : a \mapsto (a_1, \ldots, a_k)$$

  is the so-called *(Chinese) reminder map*.
- The previous theorem says that the Chinese reminder map θ is a ring isomorphism, which respects invertible elements.

Residual Arithmetic

Notes

- The restriction of the map θ to the set $\mathbb{Z}_n^*$ is a bijection of the set $\mathbb{Z}_n^*$ to the set $\mathbb{Z}_{n_1}^* \ldots\ldots \times \mathbb{Z}_{n_k}^*$, which is a group isomorphism.
- If n, m are relatively prime, then $\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$.

Consequence

Let $n = \prod_{i=1}^{k} p_i^{e_i}$, where p_i are distinct primes, so their powers are pairwise relatively prime.
If we want to count in $\mathbb{Z}_n$, we can count just in the corresponding $\mathbb{Z}_{p_i^{e_i}}$ and then use the Chinese reminder isomorphism.

Residual arithmetic

Residual arithmetic

We want to count numbers in the range $-M \leq c < M$, or respectively with numbers in the range $0 \leq c < 2M$.

We choose a family of pairwise relatively prime numbers $n_1, \dots, n_k$ such that $n = \prod_{i=1}^{k} n_i > 2M$ (just a little larger). We compute the universal coefficients q_i, $1 \leq i \leq k$ for this family.

We perform all calculations with reminders in each Z_{n_i} and finally the result in Z_n is obtained using the Chinese reminder theorem. If we know that the results are between $-M$ and M, then $M \leq c < 2M$ corresponds to $-M \leq c - n < 0$.

Residual arithmetic

Exercise

In Z_{1980}, compute $a \cdot b$, a^b, a^{-1} for the numbers
$a = 31313131313$, $b = 123456789$.

We know that $1980 = 4 \cdot 5 \cdot 9 \cdot 11$, so $\mathbb{Z}_{1980} \cong \mathbb{Z}_4 \times \mathbb{Z}_5 \times \mathbb{Z}_9 \times \mathbb{Z}_{11}$.
The q_{n_i} universals for this disjoint set of numbers are
$q_4 = 1485$, $q_5 = 396$, $q_9 = 1540$, $q_{11} = 540$.

$\theta(a) = (1, 3, 5, 2)$, $\theta(b) = (1, 4, 0, 5)$.

$\theta(a \cdot b) = (1 \cdot 1, 3 \cdot 4, 5 \cdot 0, 2 \cdot 5) = (1, 2, 0, -1)$.
Hence $a \cdot b = 1q_4 + 2q_5 + 0q_9 - 1q_{11} = 1737$ in $\mathbb{Z}_{1980}$.

$\theta(a^b) = (1^b, 3^b, 5^b, 2^b) = (1^1, 3^1, 5^3, 2^9) = (1, 3, -1, 6)$, we have used the Euler-Fermat's theorem in each $\mathbb{Z}_{n_i}$. So $a^b = 413$ in $\mathbb{Z}_{1980}$.

$\theta(a^{-1}) = (1^{-1}, 3^{-1}, 5^{-1}, 2^{-1}) = (1, 2, 2, 6)$, so $a^{-1} = 677$ in Z_{1980}.

Residual arithmetic

Divisibility theorems

Let $a = \sum_{i=0}^{k} a_i \cdot 10^i$, where a_i are digits. Then

- $a \equiv \sum_{i=0}^{k} a_i \pmod{3}$, respectively $\pmod{9}$
- $a \equiv \sum_{i=0}^{k} (-1)^i a_i \pmod{11}$

Let $a = \sum_{i=0}^{k} t_i \cdot 1000^i$, where t_i are triples of digits. Then

- $a \equiv \sum_{i=0}^{k} (-1)^i t_i \pmod{7}$, respectively $\pmod{13}$

Asymptotic notation

Definition

Let f and g be real functions and $g(x) \geq 0$ (both functions could be defined and g non-negative only "for all sufficiently large x").

- $f \in O(g)$ when there exist $c > 0$ and $x_0 \in \mathbb{R}$ such that for all $x \geq x_0$, $|f(x)| \leq cg(x)$.
- $f \in \Omega(g)$ when there exist $c > 0$ and $x_0 \in \mathbb{R}$ such that for all $x \geq x_0$, $f(x) \geq cg(x)$.
- $f \in \Theta(g)$ when there exist $c, d > 0$ and $x_0 \in \mathbb{R}$ such that for all $x \geq x_0$, $dg(x) \leq f(x) \leq cg(x)$.

Asymptotic notation

Definition

Let f and g be real functions and $g(x) \geq 0$ (both functions could be defined and g non-negative only "for all sufficiently large x").

- $f \in o(g)$ when for every $c > 0$ there exists $x_0 \in \mathbb{R}$ such that for all $x \geq x_0$ there is $|f(x)| \leq cg(x)$.
- $f \in o(g)$ when $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$.
- $f \sim g$ (asymptotically equivalent) when $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1$.

Number representation

Number length

The length of an integer a is the number of bits in the binary representation of the absolute value $|a|$, i.e.

- $\mathrm{len}(a) = \lfloor \log_2 |a| \rfloor + 1$ if $a \neq 0$
- $\mathrm{len}(a) = 1$ if $a = 0$

Number representation

Large integers representation

Large integers are stored in a computer memory as a vector of words of length $\mathrm{len}(B)$ together with a sign bit:

$$a = \pm \sum_{i=0}^{k-1} a_i B^i = \pm(a_{k-1}, \ldots, a_0)_B$$

Then $\mathrm{len}(a) = k\,\mathrm{len}(B) = O(k)$.

For example, in the languages *C* or *Java* for 32−bit computers $B = 2^{15}$ is used for a type *Integer*.

Basic operations in $\mathbb{Z}$

Statement

Let a, b be integers. Suppose that adding two bits or multiplying two bits takes one unit of time.

- $a \pm b$ takes $O(\mathrm{len}(a) + \mathrm{len}(b))$ time.
- $a \cdot b$ takes $O(\mathrm{len}(a)\,\mathrm{len}(b))$ time.
- If $b \neq 0$, $a = qb + r$, we can find the quotien q and the remainder r in $O(\mathrm{len}(b)\,\mathrm{len}(q))$ time.
  Thus $\mathrm{len}(a) - \mathrm{len}(b) - 1 \leq \mathrm{len}(q) \leq \mathrm{len}(a) - \mathrm{len}(b) + 1$.
- Multiplying a or dividing a by a power of 2^n takes $O(\mathrm{len}(a))$ time, since it's just shifting bits left or right.

Basic operations in $\mathbb{Z}$

Faster multiplication

- A classical algorithms for multiplying two numbers of length l in $O(l^2)$ time is not the fastest one. But it is sufficient for our estimate of time complexity of algorithms (we will have an upper bound of it).
- The Karatsuba's algorithm for multiplying two numbers of length l takes time $O(l^{\log_2(3)})$, while $\log_2(3) \doteq 1.58$.
- When counting with large numbers represented in the $B = 2^{15}-$ary system, multiplication of two words of length 15 takes place within a single $32-$bits word. We can assume that it takes one unit of time. Then the multiplicative constant in the time estimate will be $\frac{1}{B}$ times smaller. The choice of B does not affect theoretical calculation, but it plays an important role in practice.

Basic operations in $\mathbb{Z}_n$

Statement

Let a, b be numbers from $\mathbb{Z}_n$ ($0 \leq a, b < n$), an exponent $e \in \mathbb{N}$. We perform operations in $\mathbb{Z}_n$ and a result should be in the range $0 \leq c < n$.

- $a \pm b$ is computed in time $O(\mathrm{len}(n))$.
- $a \cdot b$ is computed in time $O(\mathrm{len}(n)^2)$.
- a^e is computed in time $O(\mathrm{len}(e)\,\mathrm{len}(n)^2)$ by the repeated squaring algorithm.
- If $\gcd(a, n) = 1$, then a^e is computed in time $O(\mathrm{len}(e)\,\mathrm{len}(n) + \mathrm{len}(n)^3)$ by the repeated squaring algorithm after using the Euler-Fermat's theorem.
- If $\gcd(a, n) = 1$, then a^{-1} in $\mathbb{Z}_n$ is computed in time $O(\mathrm{len}(n)^3)$ using the repeated squaring algorithm.

Time complexity of Euclidean algorithm

The Euclidean algorithm computes $\gcd(a, b)$, where $a \geq b > 0$.

- The number of divisions with remainder is $O(\operatorname{len}(b))$.
- The rough estimate of the total time is $O(\operatorname{len}(b)^2 \operatorname{len}(a))$.
- Moreover it can be proved that the Euclidean algorithm only needs $O(\operatorname{len}(b) \operatorname{len}(a))$ time.

The extended Euclidean algorithm computes $\gcd(a, b)$ together with $s, t \in \mathbb{Z}$ such that $sa + tb = \gcd(a, b)$.

- The extended Euclidean algorithm needs $O(\operatorname{len}(b) \operatorname{len}(a))$ time.
- If $\gcd(a, n) = 1$, then a^{-1} in $\mathbb{Z}_n$ is computed in time $O(\operatorname{len}(n)^2)$ by the extended Euclidean algorithm.

Time complexity of residual arithmetic

We count with integers a, b and we expect results in the range from $-M$ to M, or from 0 to $2M$ respectively.

- We choose a set of "small" pairwise relatively prime numbers, usually primes $p_1, \ldots, p_k$ such that $n = \prod_{i=1}^{k} p_i > 2M$.
  Say all primes $p_i < 2^C$, where C is a constant.
  We will count residually using the Chinese reminder theorem.
- The universal coefficients q_i, $1 \leq i \leq k$, for the Chinese remainder theorem can be computed in time $O(\mathrm{len}(n)^2)$, and $\mathrm{len}(q_i) \simeq \mathrm{len}(n)$. But of course, we calculate these coefficients only once!

Time complexity of residual arithmetic

- Reminders a_i, b_i of numbers a, b modulo each p_i, $1 \leq i \leq k$, are computed in time $O(C\,\mathrm{len}(n)) = O(\mathrm{len}(n))$.
- Arithmetic operations in $\mathbb{Z}_{p_i}$ take a constant time: $a_i \pm b_i$, $a_i \cdot b_i$, or a_i^r, a_i^{-1}, if $\gcd(a_i, n) = 1$, $r < p_i$, are computed in a time at most $O(C^3) = O(1)$ in each Z_{p_i}.
- A solution of the corresponding residue systems for $a \pm b$, $a \cdot b$, or a^s, a^{-1} in $\mathbb{Z}_n$, if $\gcd(a, n) = 1$, contains in counting a corresponding linear combination of the coefficients q_i performed in $\mathbb{Z}_n$ which takes time $O(kC\,\mathrm{len}(n)) = O(\mathrm{len}(n))$.
- Residual counting (with precomputed q_i's) works in a linear time with the multiplicative constant k (because C is small).

Time complexity of residual arithmetic

Example

The product of all primes smaller than 2^{16} is approximately $2^{90\ 000}$. We can residually multiply numbers which have less than 45 000 bits in a linear time.
An estimate for a multiplicative constant: there is $k \doteq 5000$ primes up to 2^{16}, multiplying remainders within a 32−bit word takes a unit of time. So it is roughly 9 times faster than a quadratic time.

Time complexity of counting modulo n

Literature

- Velebil: Discrete mathematics. Chapter 3.4.
  ftp://math.feld.cvut.cz/pub/velebil/y01dma/dma-notes.pdf
- Shoup: A Computational Introduction to Number Theory and Algebra. Chapters 2.4-7, 3.1-4, 4.1-4.
  http://shoup.net/ntb/