Mathematical Background II

Computer and Network Security

Emilio Coppa

Euler's Phi Function (or Totient Function)

Given: $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$

Q. How many numbers in it are coprime (relatively prime) to m, i.e., $\Phi(m)$?

Remark. A number a is coprime to a number m if: $gcd(a, m) = 1$

A naive approach is to check: $gcd(a, m) \stackrel{?}{=} 1 \; \forall \, a \in \mathbb{Z}_m$

E.g., Let's consider $\mathbb{Z}_6$:

$gcd(0, 6) = 6$
$gcd(1, 6) = 1$ ✓
$gcd(2, 6) = 2$
$gcd(3, 6) = 3$
$gcd(4, 6) = 2$
$gcd(5, 6) = 1$ ✓

$\Rightarrow \Phi(6) = 2$

Euler's Phi Function (2)

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8	16	6	18	8	12

Efficient way of computing Euler's Phi Function

$$\Phi(m) = \prod_{i=1}^{m} (p_i^{e_i} - p_i^{e_i - 1})$$

where:

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n}$$

p_i : prime number

e_i : positive integer

Hence it is to compute Euler's Phi Function if we have a factorization of m.

E.g.

$$m = 240, \Phi(240) = ?$$

$$m = 2^4 \cdot 3 \cdot 5 \Rightarrow \Phi(240) = (2^4 - 2^3)(3^1 - 3^0)(5^1 - 5^0) = 64$$

Fermat's Little Theorem

Given an integer a and a prime p then:

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

Hence, also:

$$a \cdot a^{p-2} \equiv 1 \bmod p$$
$$a^{-1} \equiv a^{p-2} \bmod p$$

We can use this theorem in some cases to compute the multiplicative inverse, instead of using EEA.

E.g., $$a^{p-2} = 2^5 = 32 \equiv 4 \bmod 7$$

4 is indeed the multiplicative inverse of 2 modulo 7

Euler's Theorem

Let a and m be integers with gcd(a, m) = 1 then:

$$a^{\Phi(m)} \equiv 1 \bmod m$$

Remark. if p is prime then $\Phi(p) = p - 1$ hence we get the Fermat's Little Theorem

E.g., m=12, a=5:

$$\Phi(12) = \Phi(2^2 \cdot 3) = (2^2 - 2^1)(3^1 - 3^0) = 4$$

$$5^{\Phi(12)} = 5^4 = 25^2 = 625 \equiv 1 \bmod 12$$

Example

Compute:

$$5^{200} \bmod 23$$

Since 23 is prime then:

$$\Phi(23) = 22$$

$$200 = 9 \cdot 22 + 2$$

$$5^{200} = (5^{22})^9 \cdot 5^2$$

$$5^{\Phi(23)} = 5^{22} \equiv 1 \bmod 23$$

$$5^{200} \equiv 1^9 \cdot 5^2 \bmod 23 \equiv 25 \bmod 23 \equiv 2 \bmod 23$$

Example

Compute:

$$2^{280} \bmod 251$$

Since 251 is prime then:

$$\Phi(251) = 250$$

$$280 = 1 \cdot 250 + 30$$

$$2^{280} = 2^{250} \cdot 2^{30}$$

$$2^{\Phi(251)} = 2^{250} \equiv 1 \bmod 251$$

$$2^{280} \bmod 251 = 1 \cdot 2^{30} \bmod 251$$

$$2^{30} = 2^{10} \cdot 2^{10} \cdot 2^{10} = 1024 \cdot 1024 \cdot 1024 \equiv 20^3 \bmod 251 = 219 \bmod 251$$

Chinese Remainder Theorem (CRT)

Given pairwise coprime positive integers (n_1, n_2, ..., n_k) and an arbitrary integers (a_1, a_2, ..., a_k), then the system:

$$
\begin{aligned}
x &\equiv a_1 \mod n_1 \\
x &\equiv a_2 \mod n_2 \\
&\cdots \\
x &\equiv a_k \mod n_k
\end{aligned}
$$

has a unique solution modulo $N = n_1 * n_2 * \ldots * n_k$. Check this video to see some examples that give you an intuition behind the proof of CRT.

Chinese Remainder Theorem: (2)

Steps:

1. Compute: $N = n_1 \cdot n_2 \cdot \ldots \cdot n_k$
2. Compute: $\forall i \in [1, k] : M_i = N / n_i$
3. Find inverses: $\forall i \in [1, k] : M_i^{-1} \bmod n_i$
4. Solve equation: $x = \sum_{i=1}^{k} (a_i \cdot M_i \cdot M_i^{-1}) \bmod N$

Video with a step-by-step example.

Chinese Remainder Theorem (2)

Given p and q coprime, if:

$$x \equiv a \bmod p$$
$$x \equiv a \bmod q$$

then:

$$x \equiv a \bmod p \cdot q$$

We can also exploit the viceversa argument. This can be used to “split” a modular problem on pq over two “simpler” modular problems on p and q. This is used for speeding up decryption with RSA but also in some attacks to RSA.

Group

A group G is a set of elements and one group operator "$\circ$" such that:

1. closure: $\forall a, b \in \mathrm{G} : a \circ b \in \mathrm{G}$
2. associativity $\forall a, b, c \in \mathrm{G} : (a \circ b) \circ c = a \circ (b \circ c)$
3. neutral element $\exists\, 1 \in \mathrm{G}$ such that $\forall a \in \mathrm{G} : a \circ 1 = a$
4. inverse element $\forall a \in \mathrm{G}\,, \exists\, a^{-1} \in \mathrm{G} : a \circ a^{-1} = 1$

If abelian group then also:

5. commutative $\forall a, b \in \mathrm{G} : a \circ b = b \circ a$

Order of a group

Given a finite group G (i.e., a group with a finite number of elements), the number of elements in the group (cardinality) is called “order of the group”:

$$|G| = \text{"order of the group"}$$

Example

E.g., $\mathbb{Z}_9 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ is not a multiplicative group since

$$gcd(0, 9) = 9 \neq 1$$
$$gcd(3, 9) = 3 \neq 1$$
$$gcd(6, 9) = 3 \neq 1$$

if we remove the "problematic" elements, then can define: $\mathbb{Z}_9^* = \{1, 2, 4, 5, 7, 8\}$

order of $\mathbb{Z}_9$ is 9
order of $\mathbb{Z}_9^*$ is 6

$\mathbb{Z}_m^*$ is the set of positive numbers smaller than m that are relatively coprime to m. The cardinality is thus equal to Phi(m).

Order of an element

Given $\mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

what happens if we compute all the powers of a=3?

$$a^1 = 3 \longrightarrow a^6 = a^5 \cdot a = 3$$

$$a^2 = a^1 \cdot a = 9 \longrightarrow a^7 = a^6 \cdot a = 9$$

$$a^3 = a^2 \cdot a = 27 \equiv 5$$

$$a^4 = a^3 \cdot a = 15 \equiv 4$$

$$a^5 = a^4 \cdot a = 12 \equiv 1$$

We are repeating the results... as in a “circle” with only five values. Hence:

$$3^{781225} \bmod 11 = ?$$

gives a result that is one of these five values.

Order of an element

The order ord(a) of an element a in a group G is the smaller positive integer k such that:

$$a^k = \underbrace{a \circ a \circ \ldots \circ a}_{k \text{ times}} = 1,$$

where 1 is the neutral element for the group operator of G.

E.g., ord(3)=5 in $\mathbb{Z}_{11}^*$

Cyclic Group

A group G which contains an element a with maximum order ord(a) = |G| is said to be cyclic.

Elements with maximum order are called primitive elements or generators.

E.g., ord(3)=5 in $\mathbb{Z}_{11}^*$ since $|\mathbb{Z}_{11}^*| = 10$ then 3 is NOT a generator of this group.

ord(2)=10 in $\mathbb{Z}_{11}^*$ since $|\mathbb{Z}_{11}^*| = 10$ then 2 is a generator of this group.

Remark. For every prime p, it can be proved that the abelian group $(\mathbb{Z}_p^*, \cdot)$ is a cyclic group, i.e., multiplicative group of every prime field is cyclic.

Properties of cyclic groups

Let G a finite cyclic group, then for every a in G it holds:

1. $a^{|G|} = 1$ (generalization Fermat's Little Theorem)
2. ord(a) divides |G|

E.g., ord(a) of an element a in $\mathbb{Z}_{11}^*$ must be in {1, 2, 5, 10}

How many primitive elements do we have?

Let G be a finite cyclic group then:

1. The number of primitive elements of G is Φ (|G|): any element that is coprime with |G| is a generator.
2. If |G| is prime, then all elements a != 1 $\in$ G are primitive.

Hence working with a group with prime cardinality makes it easy to find primitive elements.

Discrete Logarithm Problem (DLP) on $\mathbb{Z}_p^*$

Cyclic groups are essential for building algorithms that exploit the DLP problem in $\mathbb{Z}_p^*$

Given the finite cyclic group $\mathbb{Z}_p^*$ of order p - 1 and a primitive element a from the group and another element b from the group. The DLP is the problem of determining the integer $1 \leq x \leq p - 1$ such that:

$$a^x \equiv b \bmod p$$

x must exists since a is a generator.

Computing DL modulo a prime is a very hard problem if the parameters are sufficiently large. In practice, to make it even more robust against some attacks (e.g., Pohlig-Hellman attack based on CRT), we consider a subgroup H of $\mathbb{Z}_p^*$ such that |H| is a prime number.

Cyclic Subgroups

- Let $(G, \circ)$ be a cyclic group then every element $a \in G$ with $ord(a) = s$ is the primitive element of a cyclic subgroup with s elements.

- [Lagrange's Theorem] Let H be a subgroup of G. Then $|H|$ divides $|G|$.

- Let G be a finite cyclic group of order n and let g be a generator of G. Then for every integer k that divides n there exists exactly one cyclic subgroup H of G of order k. This subgroup is generated by $g^{n/k}$. H consists exactly of the elements $a \in G$ which satisfy the condition $a^k = 1$. There are no other subgroups.

Discrete Logarithm Problem (DLP)

DLP can be generalized to other cyclic groups (e.g., additive group), but keep in mind that DLP is not always hard to solve for ALL possible cyclic groups.

Integer Factorization Problem (IFP)

Another hard-to-solve problem used in cryptography is IFP:

given an integer n, we want to find integer a and b such that n = a*b

Integer factorization is believed to very hard when n is very large and a and b are required to be prime numbers (i.e., prime factorization).

Exponentiation

How to compute x^a? e.g., x^4

1. naive approach: (((x * x) * x) * x) [4 multiplications]
2. faster approach: ((x * x) * (x * x)) [2 multiplications (with caching on x^2)]

E.g., $x^{2^{1024}}$

1. naive approach: $2^{1024} - 1$ multiplications DOES NOT SCALE
2. faster approach: $log_2 2^{1024} = 1024$ multiplications OK

Problem: faster approach works only when exponent is a power of 2.

Square-and-Multiply (s-a-m) Algorithm

x^{26} :

$x \cdot x = x^2$	square
$x^2 \cdot x = x^3$	multiply
$x^3 \cdot x^3 = x^6$	square
$x^6 \cdot x^6 = x^{12}$	square
$x^{12} \cdot x = x^{13}$	multiply
$x^{13} \cdot x^{13} = x^{26}$	square

How to choose when to “square” and when to “multiply”?

The order of square and multiply can be derived by looking at the binary representation of the exponent:

$$x^{26} = x^{11010_2}$$

Square-and-Multiply (s-a-m) Algorithm (2)

Our goal is to obtain (from left to right) exponent 11010 starting with exponent 1:

$$(x^{1_2})^2 = x^{10_2}$$
$$(x^{10_2}) \cdot x = x^{11_2}$$
$$(x^{11_2})^2 = x^{110_2}$$
$$(x^{110_2})^2 = x^{1100_2}$$
$$(x^{1100_2}) \cdot x = x^{1101_2}$$
$$(x^{1101_2})^2 = x^{11010_2}$$

Square-and-Multiply (s-a-m) Algorithm (3)

Idea: scan bits of the exponent from left to right

1. in every iteration we square
2. if current bit is one, then we multiply by x

Complexity. Given an exponent e with a bit length of t + 1 bits then we have to perform t square operations and, on average, 0.5t multiply operations. Hence, the average cost is 1.5t.

Square-and-Multiply (s-a-m) Algorithm (4)

```
static long fastExp(int base, int exp) {
    long f = 1;
    long b = base;
    while(exp > 0) {
        int lsb = 0x1 & exp;
        exp >>= 1;
        if(lsb) f *= b;
        b *= b;
    }
    return f;
}
```

Credits

These slides are based on material from:

- Slides of Prof. D'Amore from CNS 2019-2020
- Christof Paar and Jan Pelzl. Understanding Cryptography: A Textbook for Students and Practitioners. Springer. http://www.crypto-textbook.com/
- Wikipedia (english version)