

BASIC NUMBER THEORY, GROUPS AND FINITE FIELDS

TTM4135 - Lecture 1

Tjerand Silde

05.01.2026

Motivation

- ▶ Cryptography uses mathematics, computer science and engineering
- ▶ Mostly the mathematics is *discrete mathematics* because cryptography deals with finite objects such as alphabets and blocks of characters
- ▶ We therefore look at modular arithmetic, which only deals with a finite number of values, and how to compute with these numbers
- ▶ Understanding the algebraic structure of finite objects helps to build useful cryptographic properties

Contents

Basic Number Theory

Groups

Finite Fields

Boolean Algebra

Factorization

- ▶ Let \mathbb{Z} denote the set of all integers
- ▶ For a and b in \mathbb{Z} , we say that a divides b (or a is a factor of b , or write $a|b$) if there exists k in \mathbb{Z} such that $ak = b$
- ▶ An integer $p > 1$ is said to be a *prime number* (or simply a *prime*) if its only positive divisors are 1 and p
- ▶ We can test for prime numbers by trial division (up to the square root of the number being tested)
- ▶ In a later lecture we will look at a more efficient way to check for primality

Basic Properties of Factors

1. If a divides b and a divides c , then a divides $b + c$
2. If p is a prime and p divides ab , then p divides a or b

Euclidean division

For a and b in \mathbb{Z} , $a > b$, there exist unique q and r in \mathbb{Z} such that:

$$a = bq + r$$

where $0 \leq r < |b|$.

Greatest common divisor (GCD)

The value d is the GCD of a and b , written $\gcd(a, b) = d$, if the following hold:

1. d divides a and b
2. if c divides a and b then c divides d
3. $d > 0$

We say that a and b are *relatively prime*, or *co-prime*, if $\gcd(a, b) = 1$

Euclidean algorithm

One can find $d = \gcd(a, b)$ as following, where q_i is the quotient and r_i is the remainder in equation i below:

$$a = bq_1 + r_1, \text{ for } 0 < r_1 < b$$

$$b = r_1q_2 + r_2, \text{ for } 0 < r_2 < r_1$$

$$r_1 = r_2q_3 + r_3, \text{ for } 0 < r_3 < r_2$$

\vdots

$$r_{k-2} = r_{k-1}q_k + r_k, \text{ for } 0 < r_k < r_{k-1}$$

$$r_{k-1} = r_kq_{k+1}, \text{ where } r_{k+1} = 0$$

Then $d = r_k = \gcd(a, b)$.

Back substitution (extended Euclidean algorithm)

- ▶ By *back substitution* in the Euclidean algorithm we can find integers x and y where

$$ax + by = d = r_k.$$

- ▶ Starting with the penultimate line in the algorithm, $r_{k-2} = r_{k-1}q_k + r_k$, we can compute

$$r_k = r_{k-2} - r_{k-1}q_k.$$

Then we replace r_{k-1} in this equation from the next line up,

$r_{k-1} = r_{k-3} - r_{k-2}q_{k-1}$ to get

$$\begin{aligned} r_k &= r_{k-2} - (r_{k-3} - r_{k-2}q_{k-1})q_k \\ &= r_{k-2}(1 + q_{k-1}q_k) - r_{k-3}q_k \end{aligned}$$

- ▶ Now we can use this equation to replace r_{k-2} from the line before that, and continue in the same way.
- ▶ Finally replacing r_1 by $r_1 = a - bq_1$ from the first line gives us r_k in terms of a multiple of a and a multiple of b .
- ▶ We will be particularly interested in the case where $r_k = d = 1$.

Modular arithmetic

Definition

b is a residue of a modulo n if $a - b = kn$ for some integer k .

$$a \equiv b \pmod{n} \iff a - b = kn.$$

Given $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$ and integer l , then

1. $a + c \equiv b + d \pmod{n}$
2. $ac \equiv bd \pmod{n}$
3. $la \equiv lb \pmod{n}$

Note

This means we can always reduce the inputs modulo n *before* performing multiplication or addition.

Contents

Basic Number Theory

Groups

Finite Fields

Boolean Algebra

Groups

A *group* is a set G with a binary operation \cdot satisfying the following:

- ▶ Closure: $a \cdot b \in G$ for all $a, b \in G$
- ▶ Identity: there exists an element, 1 , so that $a \cdot 1 = 1 \cdot a = a$ for all $a \in G$
- ▶ Inverse: for all $a \in G$ there exists an element, b , so that $a \cdot b = 1$
- ▶ Associative: for all $a, b, c \in G$, we have that $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

We will only be looking at commutative (or **abelian**) groups which satisfy also:

- ▶ Commutative: for all $a, b \in G$, we have that $a \cdot b = b \cdot a$

Cyclic groups

- ▶ The *order of a group* G , often written $|G|$, is the number of elements in G
- ▶ We write g^k to denote repeated application of g using the group operation — for example $g^3 = g \cdot g \cdot g$. The *order of an element* $g \in G$, often written $|g|$, is the smallest (non-zero!) integer k with $g^k = 1$
- ▶ A group element g is a *generator* for G if $|g| = |G|$
- ▶ A group is *cyclic* if it has a generator

Cyclic groups are important in cryptography because if we construct a group G with large order then we can be sure that a generator g can also take on the same large number of values.

Computing inverses modulo n

- ▶ The modular inverse of a modulo n , if it exists, is a value x such that $ax \equiv 1 \pmod{n}$ and is written $a^{-1} \pmod{n}$.
- ▶ In cryptosystems we often need to find inverses so that we can decrypt, or undo, certain operations.

Theorem

Let $0 < a < n$. Then a has an inverse modulo n if and only if $\gcd(a, n) = 1$.

Modular inverses using Euclidean algorithm

- ▶ We can use the Euclidean algorithm to efficiently find the inverse of a
- ▶ Remember that we want to solve for x , given a : $ax \equiv 1 \pmod{n}$
- ▶ Since $\gcd(a, n) = 1$ we can find $ax + ny = 1$ for integers x and y by Euclidean algorithm. Therefore:

$$ax = 1 - ny$$

$$ax \equiv 1 \pmod{n}$$

Groups of prime modulus: \mathbb{Z}_p^*

- ▶ Let \mathbb{Z}_p denote the set of positive integers less than p : $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$.
- ▶ A complete set of integers modulo any prime p with the zero removed forms a group under multiplication denoted \mathbb{Z}_p^* .
- ▶ Some useful properties:
 - ▶ The order of \mathbb{Z}_p^* is $p-1$
 - ▶ \mathbb{Z}_p^* is cyclic
 - ▶ \mathbb{Z}_p^* has many generators in general

Finding a generator of \mathbb{Z}_p^*

- ▶ A *generator* of \mathbb{Z}_p^* is an element of order $p - 1$
- ▶ A general theorem of algebraic groups (Lagrange) implies that the order of any element in a group must exactly divide $p - 1$
- ▶ To find a generator of \mathbb{Z}_p^* we can choose a value g and test it as follows:
 1. compute all the distinct prime factors of $p - 1$ and call them f_1, f_2, \dots, f_r
 2. then g is a generator as long as $g^{(p-1)/f_i} \neq 1 \pmod{p}$ for $i = 1, 2, \dots, r$

Groups of composite modulus: \mathbb{Z}_n^*

- ▶ For any n , which may or may not be prime, we can define \mathbb{Z}_n^* to be the group of integers modulo n which have an inverse under multiplication
- ▶ \mathbb{Z}_n^* is a group but is not cyclic in general
- ▶ Finding the order of \mathbb{Z}_n^* is difficult in general

Contents

Basic Number Theory

Groups

Finite Fields

Boolean Algebra

Fields

A *field* is a set \mathbb{F} with two binary operations $+$ and \cdot , satisfying the following:

- ▶ \mathbb{F} is a commutative group under the $+$ operation, with identity element 0
- ▶ $\mathbb{F} \setminus \{0\}$ is a commutative group under the \cdot operation
- ▶ Distributive: for all $a, b, c \in \mathbb{F}$:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

Finite fields

- ▶ For secure communications we are usually only interested in fields with a finite number of elements
- ▶ A famous theorem says that finite fields exist of size p^n for any prime p and positive integer n , and that no finite field exists of other sizes
- ▶ The most interesting cases for us are fields of size p for a prime p and fields of size 2^n for some integer n
- ▶ These fields are often called Galois Fields and are denoted $GF(p^n)$

The finite field $GF(p)$

- ▶ We often write \mathbb{Z}_p instead of $GF(p)$
- ▶ Multiplication and addition are done modulo p
- ▶ The multiplicative group is exactly \mathbb{Z}_p^*
- ▶ Later in the course we will see some public key encryption and digital signature schemes using $GF(p)$

The finite field $GF(2)$

- ▶ $GF(2)$ is the simplest field. It has only two elements.
- ▶ Addition is binary addition modulo 2. This is the same as the logical XOR (exclusive-OR) operation.
- ▶ Since there is only one non-zero element we have a trivial multiplicative group with the single element 1.
- ▶ We often use XOR in cryptography, usually written \oplus . For bit strings a and b we write $a \oplus b$ for the bit-wise XOR. For example,

$$101 \oplus 011 = 110$$

The finite field $GF(2^n)$

- ▶ Arithmetic in these fields can be considered as polynomial arithmetic where the field elements are polynomials with binary coefficients.
- ▶ This allow us to equate any n-bit string with a polynomial in a natural way: for example the eight-bit string 00101101 $\leftrightarrow x^5 + x^3 + x^2 + 1$.
- ▶ The field can be represented by use of a *primitive polynomial* $m(x)$.
- ▶ Addition and multiplication is defined by polynomial addition and multiplication modulo $m(x)$ and 2^n .
- ▶ Polynomial division can very efficient in hardware using shift registers.

Arithmetic in $GF(2^8)$

- ▶ This field is used for calculations in the commonly used AES block cipher.
- ▶ To add two strings we add their coefficients modulo 2 (exclusive or).
- ▶ Multiplication is done with respect to a primitive generator polynomial which for AES is chosen as:

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

- ▶ To multiply two strings we multiply them as polynomials and then take their remainder after dividing by $m(x)$.

Contents

Basic Number Theory

Groups

Finite Fields

Boolean Algebra

Boolean values

- ▶ A Boolean variable x takes the values of 1 or 0 representing *true* or *false*
- ▶ A Boolean *function* is any function with range (output) in the set $\{0, 1\}$
- ▶ Boolean functions are often represented by a *truth table*
- ▶ Each row defines one possible input and the associated output value

Boolean operations

- ▶ Logical AND: equivalent to multiplication modulo 2

x_1	x_2	$Z = x_1 \wedge x_2$
1	1	1
1	0	0
0	1	0
0	0	0

- ▶ Logical OR:

x_1	x_2	$Z = x_1 \vee x_2$
1	1	1
1	0	1
0	1	1
0	0	0

Boolean operations

- ▶ Logical negation:

x	$\neg x$
1	0
0	1

- ▶ We can also write negation as $\neg x = x \wedge 1$

Questions?