Lecture 3: Number Theory and Finite Fields (Discrete Mathematics)

COSC362 Data and Network Security

Book 1: Chapter 2

Spring Semester, 2021

Motivation

- Cryptology makes use of mathematics, computer science and engineering.
- Mostly discrete mathematics because cryptology deals with finite objects such as alphabets and blocks of characters.
- Looking at modular arithmetic which only deals with a finite number of values.
- Understanding the algebraic structure of finite objects helps to build useful cryptographic properties.

Outline

Outline

Factorisation

- $\mathbb{Z} = \{\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots\}$ is the set of integers.
- Given $a, b \in \mathbb{Z}$, a divides b if there exists $k \in \mathbb{Z}$ s.t. $ak = b$.
  - a is a factor of b
  - $a|b$
- An integer $p > 1$ is a *prime* number if its only divisors are 1 and p:
  - Examples: 2, 3, 5, 11, 13, 17, 19, etc.
- Testing prime numbers p by trial numbers, up to the square root of p (i.e. $\sqrt{p}$).
- There are more efficient ways to check for primality (later in the course).

Basic Properties of Factors

- If a divides b AND a divides c, then a divides $b + c$.
- If p is a prime and p divides ab, then p divides a OR b.

Example:
$6|18$ and $6|24 \Rightarrow 6|42$
$7|42 \Rightarrow 7|3$ or $7|14$

Division Algorithm

Given $a, b \in \mathbb{Z}$, s.t. $a > b$, then there exists $q, r \in \mathbb{Z}$ s.t.

$$a = bq + r$$

where q is the *quotient* and $0 \leq r < b$ is the *remainder*. One can show that $r < a/2$.

Example:
$17 = 5 \times 3 + 2$

Greatest Common Divisor (GCD)

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

- d divides a AND b
- if c divides a and b then c divides d
- $d > 0$

a and b are *relatively prime* when $\gcd(a, b) = 1$.

Euclidean Algorithm

Finding $d = \gcd(a, b)$ as follows:

$$\begin{aligned}
a &= bq_1 + r_1 \text{ for } 0 < r_1 < b \\
b &= r_1 q_2 + r_2 \text{ for } 0 < r_2 < r_1 \\
r_1 &= r_2 q_3 + r_3 \text{ for } 0 < r_3 < r_2 \\
&\vdots \\
r_{k-3} &= r_{k-2} q_{k-1} + r_{k-1} \text{ for } 0 < r_{k-1} < r_{k-2} \\
r_{k-2} &= r_{k-1} q_k + r_k \text{ for } 0 < r_k < r_{k-1} \\
r_{k-1} &= r_k q_{k+1} \text{ with } r_{k+1} = 0
\end{aligned}$$

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

Euclidean Algorithm

Data: a, b
Result: $\gcd(a, b)$
$r_{-1} \leftarrow a$;
$r_0 \leftarrow b$;
$k \leftarrow 0$;
while $r_k \neq 0$ **do**
  $q_k \leftarrow \lfloor \frac{r_{k-1}}{r_k} \rfloor$;
  $r_{k+1} \leftarrow r_{k-1} - q_k r_k$;
  $k \leftarrow k + 1$;
end
$k \leftarrow k - 1$;
return r_k

Back Substitution

Finding integers x, y in:

$$ax + by = d = r_k$$

by using back substitution in the Euclidean algorithm.
We have from the previous slide:

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

Rewriting:

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

Back Substitution

Getting:

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

by replacing r_{k-1} in the next line UP.

Back Substitution

Using $r_k = r_{k-2}(1 + q_{k-1}q_k) - r_{k-3}q_k$ in the next line UP.

Getting r_k as a multiple of a and a multiple of b by replacing r_1 by $r_1 = a - bq_1$.

An interesting case for us is when $r_k = d = 1$.

Example

- $\gcd(17, 3) = ?$
- Euclidean algorithm:

$$\begin{aligned} 17 &= 3 \times 5 + 2 \\ 3 &= 2 \times 1 + 1 \\ 2 &= 1 \times 2 \end{aligned}$$

  So $\gcd(17, 3) = 1$
- Back substitution:

$$\begin{aligned} 1 &= 3 - 2 \times 1 \text{ from the second to last line} \\ &= 3 - (17 - 3 \times 5) \times 1 \text{ from } 2 = 17 - 3 \times 5 \\ &= 17 \times (-1) + 3 \times 6 \text{ by reordering} \end{aligned}$$

Outline

Modular Arithmetic

Definition:
b is a residue of $a \mod n$ if $a - b = kn$ for some integer k:

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

(One can also write $a = kn + b$ where n is seen as the quotient and b as the remainder.)

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

- $a + c \equiv b + d \pmod{n}$
- $ac \equiv bd \pmod{n}$
- $ka \equiv kb \pmod{n}$

N.B.: one can always reduce the inputs modulo n BEFORE performing multiplication and addition.

Notation: $a \mod n$

$b \mod n$ denotes the unique value a in the complete set $\{0, 1, \cdots, n-1\}$ of residues such that:

$$a \equiv b \pmod{n}$$

$b \mod n$ is the remainder after dividing a by n.

Residue Class

Definition:
The set $\{r_0, r_1, \cdots, r_{n-1}\}$ is a complete set of residues modulo n if for every integer a, then $a \equiv r_i \pmod{n}$ for EXACTLY one r_i:

- Numbers $0, 1, \cdots, n-1$ form a complete set of residues modulo n since $a = qn + r$ for any a (where $0 \leq r < n$).
- $\{0, 1, \cdots, n-1\}$ is denoted as $\mathbb{Z}_n$.

Outline

Groups

Definition:

A group $\mathbb{G}$ is a set with *binary operation* $\cdot$ and:

- *Closure:* $a \cdot b \in \mathbb{G}$ for $a, b \in \mathbb{G}$
- *Identity:* there is an element 1 s.t. $a \cdot 1 = 1 \cdot a = a$ for $a \in \mathbb{G}$
- *Inverse:* there is an element b s.t. $a \cdot b = 1$ for $a \in \mathbb{G}$
- *Associativity:* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for $a, b, c \in \mathbb{G}$

A group $\mathbb{G}$ is said to be *abelian* when:

- *Commutativity:* $a \cdot b = b \cdot a$ for $a, b \in \mathbb{G}$

Cyclic Groups

- The order $|\mathbb{G}|$ of a group $\mathbb{G}$ is the number of elements in $\mathbb{G}$.
- g^k denotes the repeated application of $g \in \mathbb{G}$ using the group operation $\cdot$
  Example: $g^3 = g \cdot g \cdot g$
- The order $|g|$ of $g \in \mathbb{G}$ is the smallest integer k s.t. $g^k = 1$.
- g is a generator for $\mathbb{G}$ if $|g| = |\mathbb{G}|$.
- A group is *cyclic* if it has a generator.

Cyclic groups are important in cryptography: if we construct a group $\mathbb{G}$ with a 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 inverse of a (if it exists!) is a value x s.t. $ax = 1 \pmod{n}$ and is written $a^{-1} \mod n$.
In cryptosystems, finding inverses enables to 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 the Euclidean Algorithm

Using the Euclidean algorithm (very efficient) to find the inverse of a.
Given a, we want to find the value x s.t.:

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

Thus, there is an integer y s.t. $ax = 1 + yn$.
From $\gcd(a, n) = 1$, we write $ax + ny = 1$ and find the integers x, y using back substitution.

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

$\mathbb{Z}_p^* = \{1, 2, \cdots, p-1\}$ is a complete set of residues modulo the prime p with the value 0 removed.

Properties:

- $|\mathbb{Z}_p^*| = p - 1$
- $\mathbb{Z}_p^*$ is cyclic
- $\mathbb{Z}_p^*$ has many generators (in general)

One can see $\mathbb{Z}_p^*$ as the multiplicative group of integers $1, 2, \cdots, p-1$ which have inverses modulo p.

Example: $\mathbb{Z}_5^*$

- 5 is prime
- $\mathbb{Z}_5^*$ is a group and all numbers less than 5 are in the group
- Obtaining $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$
- There are indeed $5 - 1 = 4$ elements

Finding a Generator of $\mathbb{Z}_p^*$

- A generator of $\mathbb{Z}_p^*$ is an element of order $p - 1$.
- Lagrange theorem: the order of any element must exactly divide $p - 1$.
- Finding a generator of $\mathbb{Z}_p^*$ as follows:
    1. Compute all the distinct prime factors $f_1, f_2, \cdots, f_r$ of $p - 1$.
    2. g is a generator if and only if $g^{(p-1)/f_i} \neq 1 \mod p$ for $i = 1, 2, \cdots, r$.

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

For any *n* (not prime), $\mathbb{Z}_n^*$ is a group of residues which have an inverse under multiplication.

Properties:

- $\mathbb{Z}_n^*$ is a group
- $\mathbb{Z}_n^*$ is NOT cyclic in general
- Finding its order is difficult in general (when *n* is big)

Example: $\mathbb{Z}_6^*$

- Writing out the numbers less than 6
- Removing all of those which are not coprime to 6
- Obtaining $\mathbb{Z}_6^* = \{1, 5\}$

Fields

Definition:

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

- $\mathbb{F}$ is an *abelian group* under the operation $+$, with identity element 0
- $\mathbb{F} \setminus \{0\}$ is an *abelian group* under the operation $\cdot$, with identity element 1
- *Distributivity:* $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for $a, b, c \in \mathbb{F}$

Finite Fields

- Setting up secure communications requires fields with a finite number of elements.
- (famous) Theorem: finite fields exist of size p^n for any prime p and positive integer n, and no finite field exists of other sizes.
- Interesting cases for us: fields of size p for a prime p and fields of size 2^n for some integer n.

Finite Field $GF(p)$

- $GF(p) = \mathbb{Z}_p$
- Multiplication and addition done modulo p
- Its multiplicative group is exactly $\mathbb{Z}_p^*$
- Some public key encryption and digital signature schemes use $GF(p)$ (see later)

Finite Field $GF(2)$

- $GF(2)$ is the simplest field, with 2 elements 0 and 1
- Addition modulo 2: the same as the logical XOR (exclusive-OR) operation
- Only one non-zero element: trivial multiplicative group with the single element 1
- XOR is often used in crypto, written $\oplus$:
  - For bit strings $a, b, a \oplus b$
  - Example: $101 \oplus 011 = 110$

Finite Field $GF(2^8)$

- Field used for calculations in AES (block cipher)
- Arithmetic in this field considered as polynomial arithmetic where the field elements are polynomials with binary coefficients
  - Equating any 8-bit string with a polynomial in a natural way
  - Example: $00101101 \leftrightarrow x^5 + x^3 + x^2 + 1$
- Polynomial division can be done very efficiently in hardware using shift registers

Arithmetic in $GF(2^8)$

- Adding 2 strings by adding their coefficients modulo 2 (XOR)
- Multiplication done with respect to a generator polynomial
  - for AES, $m(x) = x^8 + x^4 + x^3 + x + 1$
- Multiplying 2 strings by multiplying them as polynomials and taking their remainder after dividing by $m(x)$

Outline

Boolean Values

- Boolean variable x takes values 0 or 1 (representing *false* and *true* respectively)
- Boolean function has its output in the set $\{0, 1\}$
- Truth table used to represent boolean function

Truth Tables

Logical AND (equivalent to multiplication modulo 2):

x_1	x_2	$x_1 \wedge x_2$
1	1	1
1	0	0
0	1	0
0	0	0

Negation:

x	$\neg x$
1	0
0	1

Logical OR:

x_1	x_2	$x_1 \vee x_2$
1	1	1
1	0	1
0	1	1
0	0	0