Lecture Note 3

Mathematical Background

Sourav Mukhopadhyay

Cryptography and Network Security - MA61027

- In order to understand some of the cryptographic algorithms dealt with throughout this course, it is necessary to have some background in two areas of mathematics
  1. Number Theory.
  2. Abstract Algebra.
- New Advanced Encryption Standard (AES) relies on the subject of finite fields which forms a part of abstract algebra.
- Going to deal with Number Theory for the moment.

Number Theory

- Number theory deals with the theory of numbers and is probably one of the oldest branches of mathematics.
- It is divided into several areas including elementary, analytic and algebraic number theory.
- These are distinguished more by the methods used in each than the type of problems posed.
- Relevant ideas discussed here and include:
  - prime numbers,
  - The greatest common divisor,

- The modulus operator,
- The modular inverse,
- Euler's Theorem and Fermat's little Theorem.

Prime Numbers

- A prime number p is an integer *greater* than 1 with only two *positive* divisors, 1 and itself.
- Therefore it's entire set of divisors (i.e. its factors) consist only of four integers ± 1 and $\pm p$.
- It can be seen that 1 is *not* a prime number.
- Prime numbers are of the utmost importance to certain cryptographic algorithms and most of the techniques used will not work without them.

- Any positive integer $I \geq 2$ is either a prime or can be expressed as the product of primes.
- This is known as the **fundamental theorem of arithmetic**:

$$I = P_N^{\epsilon_N} \times P_{N-1}^{\epsilon_{N-1}} \times, \ldots, \times P_1^{\epsilon_1}, \qquad P_N > P_{N-1} > \ldots > P_1 \tag{1}$$

- Another way of looking at this would be:

$$I = \prod_S P_n^{\epsilon_n}, \qquad \epsilon_n \geq 0 \tag{2}$$

- Here S is the set of all prime numbers.

- In general most of the exponents ϵ_n will be 0.
- As a result of equations 1 or 2, any integer > 1 that is not a prime is known as a **composite number**.
- It can be seen from this and the definition of a prime number above, that 1 is neither prime nor composite.
- The first ten prime numbers are:
  $2, 3, 5, 7, 11, 13, 17, 19, 23$ and 29.

Division

- Any integer can be expressed as $n = q \times m + r$, where n, q, and r are integers, m is a positive integer and $0 \leq r < m$.
- The **remainder** (also known as **residue**) r, must be *nonnegative* (i.e. either positive or 0).
- This is seen by two restrictions:
  1. $0 \leq r < m$
  2. $q = \lfloor \frac{n}{m} \rfloor$
- The notation $\lfloor x \rfloor$ is known as the **floor** of the integer x and is the greatest integer $\leq x$.

- The notation $\lceil x \rceil$ is the **ceiling** of the integer x and is the least integer $\geq x$.
- For example, $24 \div 10$ is 2 with a remainder of 4 however, $-24 \div 10$ is -3 with a remainder of 6 and not -2 with a remainder of -4 as might be expected.
- If $r = 0$ then n is said to be a multiple of m. This is also the same as saying that m divides n, is a divisor of n or is a factor of n and the notation used to express this is $m|n$.

- **The greatest common divisor**, m_{max}, of two integers a and b is the largest *positive* integer that will divide both a and b without a remainder.
- Therefore, $m_{max}|a$, $m_{max}|b$ and $m_n|m_{max}$ for any divisor m_n of a and b.
- The notation generally used to represent this is $gcd(a, b) = m_{max}$.
- If $gcd(a, b) = 1$, this means that a and b have no common factors other than 1.
- Such pairs of integers are known as **relatively prime** or **co-prime**.

- Along with prime numbers, numbers that are relatively prime have considerable importance in cryptography as will be seen later.
- The greatest common divisor of two positive integers a and b ($gcd(a, b)$), can be determined by a procedure known as **Euclid's Algorithm**. It is based on the theorem that $gcd(a, b) = gcd(b, a \bmod b)$.
- The expression "mod" is used in modular arithmetic which is a special kind of arithmetic involving remainders as will be seen next.

Modular Arithmetic

- The symbol used ($\equiv$) is known as the **congruence symbol**.
- Modular relationships are of the form $n \equiv R \pmod{m}$ (spoken as "n is congruent to R mod m") where n and R are integers and m is a positive integer known as the **modulus**.
- If this congruence relationship holds, then it is said that n is congruent to R modulo m.
- The modulus operator (mod) produces the remainder when the integer on it's left is divided by the modulus.

- Thus, the term $(R \bmod m)$ is equal to the remainder, r, when R is divided by m.
- If two remainders are equal then it can be written that $(n \bmod m) = (R \bmod m)$ - a standard equality.
- However, if the modulus is equal on both sides of the equation, then the $\pmod m$ term can be removed from the left hand side and the equality symbol replaced with a congruence symbol (along with a slight rearrangement of the brackets).

- Assuming $n \neq r$, it would be *incorrect* to say $n = (R \bmod m)$.
- However it is *correct* to say that $n \equiv R \pmod{m}$ and this basically states that the same remainder (in this case r) results when both n and R are divided by m.
- As mentioned, the remainder r is also known as a **residue**.
- If $R = r$ (i.e. $0 \leq R < m$) then R is known as a **least residue**.
- The congruent relation $\equiv$ is an equivalence relation.

- The set of numbers congruent to some value $a \pmod{m}$ is known as a **residue class** (or a congruence class).
- As $0 \leq r < m$, this means there are m possible values of r and hence there are m possible residue classes.
- The congruence relationship $n \equiv R \pmod{m}$ is only true if $m|(n-R)$.
- If $a \equiv a_1 \pmod{m}$ and $b \equiv b_1 \pmod{m}$, then $a+b \equiv a_1+b_1 \pmod{m}$ and $ab \equiv a_1 b_1 \pmod{m}$.
- The integers modulo m, denoted Z_m is the set of integers $\{0, 1, 2, \ldots, m-1\}$.

- To understand why, it must be remembered that the integers n and R can be expressed as $q_{\{n,R\}} \times m + r_{\{n,R\}}$, where the subscript $\{n, R\}$ represents the fact that q and r will generally take on different values for n and R.
- Only if $r_n = r_R$ will $m|(n - R)$ because in this case the two remainders cancel each other in the $(n - R)$ term: $(q_n \times m + r_n - q_R \times m - r_R) = (q_n \times m - q_R \times m)$.
- Because $m|(q_n \times m)$ and $m|(q_R \times m)$ $\Rightarrow m|(q_n \times m - q_R \times m)$. If $n - R$ is not divisible by m then the notation used to represent this is $\not|$ and therefore, $m \not| (n - R)$. In this case $n \not\equiv R \pmod{m}$.

Modular Inverse

- The idea of an inverse is important both in ordinary arithmetic and modular arithmetic.
- In any set of numbers, the inverse of a number contained in that set is another number which when combined with the first under a particular operation will give the **Identity element** for that operation.
- The identity element for a particular operation is a number that will leave the original number unchanged under that operation.

- Two examples of inverses are the: (1) Additive inverse and (2) Multiplicative inverse.
- The Identity element under different operations will be different.
- Under addition it is 0, as any number added to 0 will remain unchanged.
- However, under *normal* multiplication the Identity element is 1 as any number multiplied by 1 will remain unchanged.

- In ordinary arithmetic if the number is x then the additive inverse is $-x$ and multiplicative inverse is $\frac{1}{x}$.
- The idea is the same in modular arithmetic however if x is an integer then its multiplicative inverse would not be $\frac{1}{x}$ as there is no such thing as a fraction in modular arithmetic.
- In this case it would be a number which, when multiplied by the original number, would give a result that is congruent 1 modulo m (again, m is the modulus).

- A number x can only have a multiplicative inverse if it is relatively prime to the modulus (i.e., $gcd(x, m) = 1$).
- When one number is operated on modulo some other number, it is said that the first number has been **reduced** modulo the second and the operation is called a **modular reduction**.
- Let $a \in Z_m$. The multiplicative inverse of a modulo m is an integer $x \in Z_m$ such that $ax \equiv 1 \pmod{m}$. If such an x exists, then it is unique, and a is said to be invertible. The inverse of a is denoted by a^{-1}.

Chinese Remainder Theorem (CRT)

- Suppose $m_1, m_2, \ldots, m_r$ are pairwise relatively prime, then the following system of simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_r \pmod{m_r} \end{aligned}$$

has a unique solution modulo $M = m_1 m_2 \ldots m_r$, which is given by

$$x = \sum_{i=1}^{r} a_i M_i y_i \bmod M,$$

where $M_i = M/m_i$ and $y_i = M_i^{-1} \bmod m_i$, for $1 \leq i \leq r$.

Example

- Consider the following system of simultaneous congruences

$$x \equiv 5 \pmod{7}$$
$$x \equiv 3 \pmod{11}$$
$$x \equiv 10 \pmod{13}$$

- Then we have the unique solution

$$x = 894 \bmod 1001.$$

Euler’s Theorem

- Euler’s Theorem can be stated mathematically as:

$$a^{\phi(m)} \equiv 1 \pmod{m}, \qquad gcd(a,m) = 1 \tag{3}$$

- a is any integer and m is the modulus (which, again, is restricted to being a positive integer).
- The symbol $\phi(m)$ is known as Euler’s *phi* (or **totient**) function and is the number of positive integers $\leq m$ and relatively prime to it.

- A few points should be noted about $\phi(m)$:
  - The value of $\phi(1)$ is defined as being equal to 1.
  - If p is some prime, then $\phi(p) = p - 1$ as there are $p - 1$ positive integers $< p$ and relatively prime to it.
  - If p and q are prime numbers and $n = pq$, then $\phi(n) = \phi(p)\phi(q) = (p - 1)(q - 1)$.
  - If $n = p_k^{\epsilon_k} \times P_{k-1}^{\epsilon_{k-1}} \times, \ldots, \times p_1^{\epsilon_1}$ is the prime factorization of n, then

$$\phi(n) = n\left(1 - \frac{1}{p_k}\right)\left(1 - \frac{1}{p_{k-1}}\right)\cdots\left(1 - \frac{1}{p_1}\right)$$

- As an example take $4^{10} = 1,048,576 \equiv 1 \pmod{11}$ because $11 \times 95,325 + 1 = 4^{10}$.
- If n is a product of distinct primes, and if $r \equiv s \pmod{\phi(n)}$, then $a^r \equiv a^s \pmod{n}$ for all integers a. In other words, when working modulo such an n, exponents can be reduced modulo $\phi(n)$.

Fermat's Little Theorem

- Fermat's Little Theorem is really a specific case of Euler's theorem where m is prime.
- Historically though, Fermat's little theorem was discovered long before Euler's Theorem.
- It can be stated as follows:

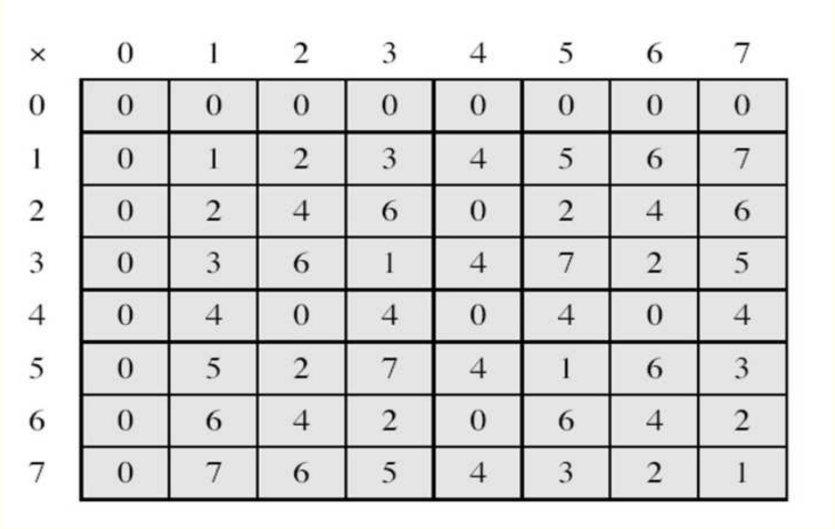
$$a^{m-1} \equiv 1 \pmod{m}, \tag{4}$$

where m is a prime and $m \not| a$

- If Euler's theorem is taken to be true, then this can also be seen to work because of the fact that $\phi(m) = m - 1$ for a prime number as mentioned above.

Extended Euclid Algorithm

- It was stated earlier that the greatest common divisor of two numbers can be found using Euclid's algorithm.
- This algorithm can be extended so that it not only finds the greatest common divisor but also calculates the inverse of some number b modulo some other number m (assuming it exists).
- For small values of m it is easy enough to find.
- However for large numbers this approach is not practical.
- Extended Euclid's algorithm allow us to find the inverse of a number b mod m assuming $\gcd(m, b) = 1$.



×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Figure 1: Arithmetic modulo 8

EXTENDED EUCLID(m,b)

```
1. $(A_1, A_2, A_3) \leftarrow (1, 0, m)$; $(B_1, B_2, B_3) \leftarrow (0, 1, b)$
2. **if** $B_3 = 0$ **return** $A_3 = \gcd(m, b)$; no inverse
3. **if** $B_3 = 1$ **return** $B_3 = \gcd(m, b)$; $B_2 = b^{-1} \bmod m$
4. $Q = \lfloor \frac{A_3}{B_3} \rfloor$
5. $(T_1, T_2, T_3) \leftarrow (A_1 - QB_1, A_2 - QB_2, A_3 - QB_3)$
6. $(A_1, A_2, A_3) \leftarrow (B_1, B_2, B_3)$
7. $(B_1, B_2, B_3) \leftarrow (T_1, T_2, T_3)$
8. Goto 2
```

- This is seen to work because $bB_2 = 1 - mB_1$ implies that $bB_2 \equiv 1 \pmod{m}$. Therefore the value of B_2 is a number when multiplied by b will give a value which is congruent to 1 modulo m (in other words it gives a value that when divided by m will leave a remainder of 1).
- We can therefore say:

$$bB_2 = 1 - mB_1, \text{i.e.,} 1 = mB_1 + bB_2 \tag{5}$$

- In other words we are trying to find two values B_1 and B_2 that solve equation 5. These values will be revealed when another value B_3 is equal to 1 in the above algorithm:

$$mB_1 + bB_2 = B_3 \tag{6}$$

- In order to find this multiplicative inverse we need to keep track of A_1, A_2 and A_3 also.
- The values T_1, T_2, T_3 are only used for temporary storage.
- Looking at steps 5 and 7 it can be seen the $B_3 \leftarrow A_3 - QB_3$ - a consequence of Euclid's algorithm and it leaves the remainder when A_3 is divided by B_3 (you are subtracting B_3 away from A_3 as many times as you can, remember $Q = \lfloor \frac{A_3}{B_3} \rfloor$).
- Throughout the algorithm, the following relationships hold:

$$mT_1 + bT_2 = T_3; mA_1 + bA_2 = A_3; mB_1 + bB_2 = B_3$$

- These equations are why the initial assignments are $(1, 0, m)$ and $(0, 1, b)$. If you work them out you will get the above for A_3 and B_3. The last equation is the one we are interested in and when $B_3 = 1$ then $B_2 = b^{-1} \bmod m$.
- For example to find the multiplicative inverse of 550 modulo 1759 we have:

Q	A_1	A_2	A_3	B_1	B_2	B_3
-	1	0	1759	0	1	550
3	0	1	550	1	-3	109
5	1	-3	109	-5	16	5
21	-5	16	5	106	-339	4
1	106	-339	4	-111	$\boxed{355}$	1

- Some topics that will be discussed are:
  - Groups
  - Rings
  - Fields
  - Polynomial arithmetic
- As well as these topics we will also be looking at an area known as **complexity theory** which allows us to determine how efficient a particular algorithm is.

Abstract Algebra

- Will only be looking at a very small subset of what this subject has to offer.
- Three main ideas here that need to be grasped:
  1. Group $\{G, \cdot\}$
  2. Ring $\{R_g, +, \times\}$
  3. Field $\{F, +, \times\}$
- Basically three different types of sets along with some operation(s).
- The classification of each set is determined by the axioms which it satisfies.

Group

- A **Group** $\{G, \cdot\}$ is a set under some operation $(\cdot)$ if it satisfies the following 4 axioms:
  1. **Closure** (A_1): For any two elements $a, b \in G$, $c = a \cdot b \in G$
  2. **Associativity** (A_2): For any three elements $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  3. **Identity** (A_3): There exists an **Identity** element $e \in G$ such that $\forall_{a \in G}$, $a \cdot e = e \cdot a = a$.
  4. **Inverse** (A_4): Each element in G has an inverse i.e. $\forall_{a \in G} \exists_{a^{-1} \in G}$, $a \cdot a^{-1} = a^{-1} \cdot a = e$.

- However it is said to be an **Abelian group** if in addition to the above the set follows the axiom:

5. **Commutativity** (A_5): For any $a, b \in G$, $a \cdot b = b \cdot a$.

Cyclic group

- **Exponentiation** is repeated application of the group operator.
- We might have a^3 and this would equal $a \cdot a \cdot a$.
- So if the operation was addition then a^3 would in fact be $a + a + a$.
- Also we have $a^0 = e$ which for an additive group is 0.
- Also $a^{-n} = (a^{-1})^n$.
- A group is said to be **cyclic** if every element of the group G is a power a^k (where k is an integer) of a fixed element $a \in G$.

- The element a is said to generate G or be a **generator** of G.
- A cyclic group is always abelian and may be finite or infinite.
- If a group has a finite number of elements it is referred to as a **finite group**.
- The **order** of the group is equal to the number of elements in the group. Otherwise, the group is an **infinite group**.

Ring

- A binary operation is a mapping of two elements into one element under some operation. For a set S we have $f : S \times S \rightarrow S$.
- A **Ring** $\{R_g, +, \times\}$ is a set with two binary operations *addition* and *multiplication* that satisfies the following axioms:
  1. **Abelian Group under addition** $(A_1 \rightarrow A_5)$: It satisfies all of the axioms for an abelian group (all of the above) with the operation of *addition*. The identity element is 0 and the inverse is denoted $-a$.

2. **Closure under multiplication** (M_1): For any two elements $a, b \in R_g$, $c = ab \in R_g$.
3. **Associativity of multiplication** (M_2): For any elements $a, b, c \in R_g$, $(ab)c = a(bc)$.
4. **Distributive** (M_3): For any elements $a, b, c \in R_g$, $a(b + c) = ab + ac$.

- It is then said to be a **commutative ring** if in addition the ring follows the axiom:

5. **Commutativity** (M_4): For any $a, b \in R_g$, $ab = ba$.

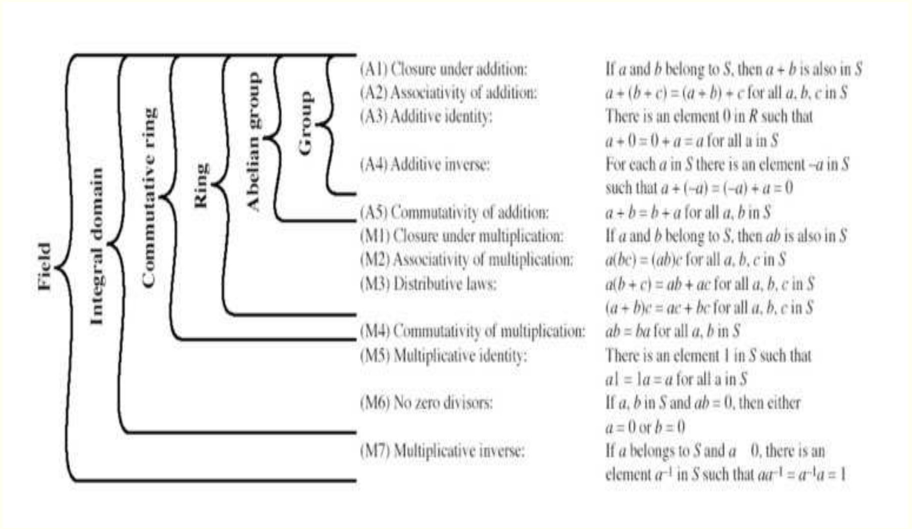
- It is an **Integral domain** if in addition the commutative ring follows the axioms:

6. **Multiplicative Identity** (M_5): There is an element 1 in R_g such that, $a1 = 1a = a$ for all a in R_g.
7. **No Zero Divisors** (M_6): If $a, b \in R_g$ and $ab = 0$ then *either* $a = 0$ *or* $b = 0$.

- A **Field** $\{F, +, \times\}$ is a set with two binary operations *addition* and *multiplication* that satisfies the following axioms:
  1. **Integral Domain** $(A_1 - M_6)$: It satisfies all of the axioms for an Integral domain (all of the above).
  2. **Multiplicative Inverse** (M_7): Each element in F (except 0) has an inverse i.e., $\forall_{a \neq 0 \in F} \exists_{a^{-1} \in F}, \; aa^{-1} = a^{-1}a = 1.$
- In ordinary arithmetic it is possible to multiply both sides of an equation by the same value and still have the equality intact.

- Not necessarily true in finite arithmetic
- In this particular type of arithmetic we are dealing with a set containing a finite number of values.
- The set of real numbers is an infinite set and is not really useful for working with on computer systems due to the limited amount of memory and processing power.
- Much easier if every operation the computer performed resulted in a finite value that was easily handled. This is where finite fields come into play.

- Closure is the property that causes the result of a binary operation on an ordered pair of a set to be a part of that set also.
- The term *ordered pair* is important as it is not generally the case that $a \cdot b = b \cdot a$.



(A1) Closure under addition:	If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition:	$a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity:	There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse:	For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$
(A5) Commutativity of addition:	$a + b = b + a$ for all a, b in S
(M1) Closure under multiplication:	If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in S $(a + b)c = ac + bc$ for all a, b, c in S
(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in S
(M5) Multiplicative identity:	There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors:	If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$
(M7) Multiplicative inverse:	If a belongs to S and $a \quad 0$, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

Figure 2: Group, Ring and Field

- To restate, Groups, Rings and Fields are all sets defined with either one or more binary operations.
- For example a set may be a group under one binary operation but not under another because it may obey the axioms $A_1 \rightarrow A_4$ under the first operation but not under the second.
- Figure 2 summarises the hierarchical structure of the group, ring and field. It can be seen that the group is defined under addition.
- Although a group is defined under operations other than addition, a ring requires that the group be defined with only addition.

Polynomial Arithmetic

- We need to understand some things about arithmetic involving polynomials before continuing.
- A **polynomial** is an *expression* of the form:

$$f(x) = a_n x^n + a_{n-1} x^{x-1} + \ldots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i \quad (7)$$

- $a_n \neq 0$,
- The **degree** of the polynomial is equal to the value of the integer $n \geq 0$.
- The **coefficients** are the set $S = \{a_n, a_{n-1}, \ldots, a_1, a_0\}$ which is known as the **coefficient set**.

- If $a_n = 1$ then the polynomial is said to be **monic**.
- If $n = 0$ then we simply have a constant known as a **constant polynomial**.
- As an example we can set $n = 8$ and $S = \{1, 0, 0, 0, 1, 1, 1, 1, 1\}$ and we get the following polynomial:

$$\sum_{i=0}^{8} a_i x^i = x^8 + x^4 + x^3 + x^2 + x + 1$$

- This polynomial has important significance for us as it is used in the AES standard as we shall see later.

- As we are going to use these for cryptographic methods we will want to do some form of arithmetic on them.
- One can do following three classes of arithmetic:
  1. Ordinary polynomial arithmetic, using the basic rules of algebra.
  2. Polynomial arithmetic in which the arithmetic on the coefficients is performed modulo p; that is the coefficients are in $Z_p = 0, 1, \ldots, p-1$.
  3. Polynomial arithmetic in which the coefficients are in Z_p and the polynomials are defined modulo a polynomial $m(x)$ having highest power, say n.

- Can we operate on polynomials using the four basic arithmetical operations of addition, subtraction, multiplication and division?
- Consider two polynomials $f(x) = x^3 + x^2 + 2$ and $g(x) = x^2 - x + 1$. If we add these we get:

$$f(x) + g(x) = x^3 + 2x^2 - x + 3$$

- This would seem to suggest we can add (and it turns out we can).
- If we subtract them:

$$f(x) - g(x) = x^3 + x + 1$$

- This would seem to suggest we can subtract (and it turns out we can).
- If we multiply them:

$$f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$$

- This would seem to suggest we can multiply (and it turns out we can).
- The order of the product polynomial is equal to the sum of the orders of the two factors. In this case $n_{prod} = n_{f(x)} + n_{g(x)} = 3 + 2 = 5$.

- What about division?
- Division requires that the set of coefficients $S = \{a_n, a_{n-1}, \ldots, a_1, a_0\}$ be a field.
- In other words S must satisfy the conditions described earlier.
- In general, division will produce a quotient and a remainder so we can write (for two polynomials $f(x)$ and $g(x)$):

$$\frac{f(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}, \text{i.e. } f(x) = q(x)g(x) + r(x) \qquad (8)$$

Galois Fields

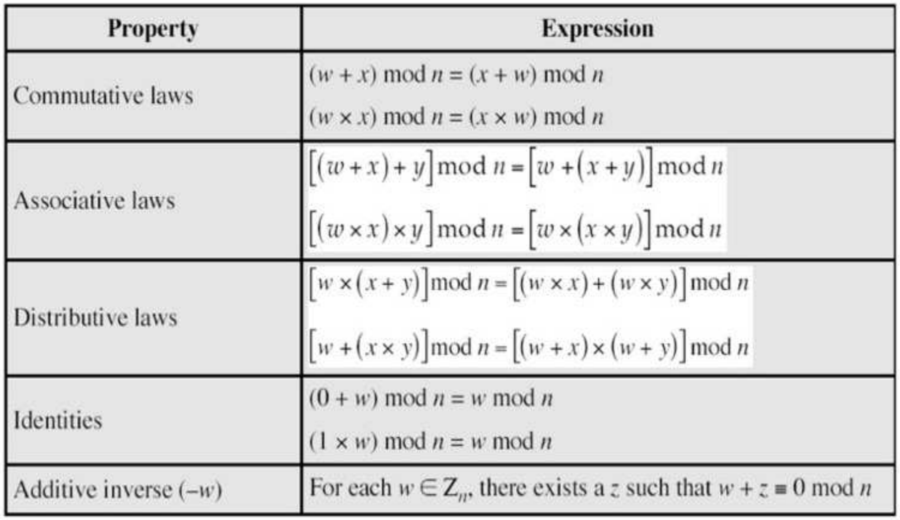
- Saw earlier that a *field* obeys axioms $A_1 \rightarrow M_7$.
- It is possible for a set with an infinite number of elements to be a field.
- For example the set of real numbers is a field under the usual arithmetical operations.
- In cryptography however we are not interested in infinite fields because they cannot be worked with in practice (due to memory limitations etc.).
- What cryptographers want instead are *finite fields*. These are simply fields with a finite number of elements.

- It turns out that the order of a finite field must be the power of a prime p^n where $n > 0$.
- The finite field of order p^n is normally written $\mathrm{GF}(p^n)$.
- The GF stands for **Galois Field** in honour of the mathematician who first studied them.
- When $n = 1$ Galois fields take on a different structure than when $n > 1$.
- We will mainly be interested in $\mathrm{GF}(p)$ for some prime p and $\mathrm{GF}(2^n)$ (2 being the main prime of interest due to computers operating in binary).

- Addition and multiplication in a Galois field are done modulo m(x), where m(x) is an irreducible polynomial of degree n.
- A polynomial m(x) over a field F is called **irreducible** if and only if m(x) cannot be expressed as a product of two polynomials both over F, and both of degree lower than that of m(x).
- By analogy to integers, an irreducible polynomial is also called a **prime polynomial**.
- As an example of a Galois field we can look at GF(2^3).

- This has the following elements: $\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$ and an irreducible polynomial is x^3+x+1.
- In binary Galois fields all polynomials are monic due to the fact that the coefficients are taken from the set $S = \{0, 1\}$.

- Earlier we stated that Z_m was the set of integers modulo m.
- If $m = p$ where p is some prime, then we have $Z_m = Z_p = \{0, 1, 2, \ldots, p - 1\}$.
- This, together with the arithmetic operations modulo p, form the finite field of order p.
- Since p is the power of a prime, this field is a Galois field.
- Table 3 shows the properties of modular arithmetic for integers in Z_m.
- If you compare this to figure 2 then it can be seen that Z_p is a commutative ring.



Property	Expression
Commutative laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive laws	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$ $[w + (x \times y)] \bmod n = [(w + x) \times (w + y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive inverse $(-w)$	For each $w \in Z_n$, there exists a z such that $w + z \equiv 0 \bmod n$

Figure 3: Properties of modular arithmetic for integeres in Z_m.

- It was seen that if some number $a \in Z_m$ is relatively prime to m then there exists a^{-1} such that $aa^{-1} \equiv 1 \pmod{m}$, i.e. the multiplicative inverse of a exists.
- However if $m = p$ for some prime p then each nonzero element in Z has a multiplicative inverse because each $a \in Z_p$ is relatively prime to p (M_5).
- Also if $ab = 0$ for $a, b \in Z_p$ then either $a = 0$ or $b = 0$.
- It can therefore be seen that Z_p is a finite field.
- This is fine for GF(p) where $n = 1$ because p is a prime but what about the case where $n > 1$? In this case p^n will not be a prime.

- In the general case, if $m = p^n$ where p is again some prime and n some integer greater than 1, m will not be prime.
- This poses problems as Z_m will not form a finite field when the arithmetical operations are done modulo m.
- Remember we need this if we wish to use division as one of our operations.
- However, if we add two restrictions to our arithmetical operations (which follow the ordinary rules of polynomial arithmetic using the basic rules of algebra) it is possible to keep Z_m as a finite field.

- These restrictions are:
    1. Arithmetic on the coefficients is performed modulo p. That is, we use the rules of arithmetic for the finite field Z_p (for GF(2^n) this is basically the XOR operation).
    2. If multiplication results in a polynomial of degree greater than $n-1$, then the polynomial is reduced modulo some irreducible polynomial $m(x)$ of degree n. That is, we divide by $m(x)$ and keep the remainder. For a polynomial f(x), the remainder is expressed as $r(x) = f(x) \bmod m(x)$.

- It can be shown that the set of all polynomials modulo an irreducible nth-degree polynomial $m(x)$ satisfies axioms $A_1 \longrightarrow M_7$ above and therefore forms a finite field.
- This is an important point because if we were working in GF(2^8) and we reduced the polynomial modulo 256 then this wouldn't form a finite field and we might have to think about using 251 as this is the closest prime to 256.
- Not efficient because might be using a processor that operates on 8 bit words which allows representation of $2^8 = 256$ values.
- Other values ($251 \longrightarrow 256$) would not be used.

Complexity theory

- All cryptographic algorithms require time and space (memory) to execute.
- Clearly what is desired is an algorithm that executes as quickly as possible with the minimum number of resources possible, however, often this is not feasible.
- **Complexity Theory** deals with the resources required during computations to solve a given problem.
- The most common resources (for example) would be *time* and *space*.

- Complexity theory allows us to work out how costly different algorithms (such as DES, RSA) are going to be.
- Also allows us to analyse attacks on cryptosystems as these will be algorithmic in nature as well.
- Using complexity theory we can determine whether a particular attack is feasible against a cryptographic algorithm and can therefore give some indication of the security of an algorithm.
- Of course one could say that the time of execution of an algorithm depends on speed of the computer on which it is running, as well as the amount of memory it has.

- This is true, so what we would like is a method of comparing different algorithms that is independent not only of the speed and amount of RAM a particular machine has, but also of the size of the input (which is normally denoted by n).
- A measure of the efficiency of an algorithm is known as its **Time Complexity**.
- The time complexity of an algorithm is defined to be $f(n)$ if for all n and all inputs of length n, the algorithm takes at most $f(n)$ steps.

- Thus for a given processor speed and a given size of input (n) the time complexity is an *upper bound* on the execution time of the algorithm.
- However, the definition is not precise:
  - The meaning of "step" is not precise: Is it a single processor machine instruction or a single high-level language machine instruction?
  - An exact formula for $f(n)$ is generally not available but that is not important. We need only an approximation and are interested primarily in the rate of change of $f(n)$ as n increases to very large values.

- There is a standard mathematical notation known as the "big O" notation which is used to characterise the time complexity of algorithms.
- By definition:-

$$\begin{aligned} f(n) = O(g(n)) \text{ iff } \exists\, a, M \in Z^+ \text{ such that} \\ |f(n)| \le a \times |g(n)| \text{ for } n \ge M \end{aligned} \tag{9}$$

- For example we might say that an algorithm runs in $O(n^2)$ time.
- Anything that takes a fixed amount of resources takes $O(1)$.

- Lets say we want to evaluate the following polynomial:

$$p(x) = a_n x^n + a_{n-1} x^{x-1} + \ldots + a_1 x + a_0$$

- Some fairly inefficient coding is shown in next slide.
- In this implementation, each term (subexpression) is evaluated separately. It could have been more efficient however. Each $a_i x^i$ requires $i + 1$ multiplications.
- Computing all n terms requires

$$\sum_{i=0}^{n} (i + 1) = \frac{(n + 2)(n + 1)}{2} \text{ multiplications.}$$

Badly written pseudocode

```
Declare $n, i, j$ integer;
Declare $a, s$ array[100] real;
Declare $x, p$ real;
Read($x, n$);
for $i = 0$ to $n$ do        \* Each of $n$ terms of the $f(x)$ *\
  {
    $s[i] = 1$;
    read(a[i]);
    for $j = 1$ to $i$ do
      {
        $s[j] = x \times s[i]$;        \* Calculate $x^i$ and store in $s[j]$*\
      }
    $s[i] = a[i] \times s[i]$;        \* Calculate $a_i x^i$ and store*\
  }
$p = 0$;
for $i = 0$ to $n$ do
  {
    $p = p + s[i]$;
  }
write(“Value of function at”, x, “is”, p “.”);
```

- The time complexity of the algorithm is :

$$f(n) = \frac{(n+2)(n+1)}{2} = \frac{n^2}{2} + \frac{3}{2}n + 1$$

- We can show that in this case $f(n) = O(n^2)$ (i.e. $g(n) = n^2$). We must find $M \leq n$ and a such that

$$|f(n)| \leq a \cdot |g(n)|$$

- If we let $M = 4$ and $a = 1$ then equation 9 is satisfied as suggested by table 1. In this case we can say that $O(f(n)) = n^2$.

n	n^2	$f(n) = \frac{n^2}{2} + \frac{3}{2}n + 1$	$g(n) = n^2$	function ok?
3	9	10	9	No
4	16	15	16	Yes
5	25	21	25	Yes
6	36	28	36	Yes
7	49	35	49	Yes

Table 1: This table shows the value of the function $f(n)$ for different n.

- In general, the "big O" notation makes use of the term that grows fastest:
  - $O(ax^7 + 3x^3 + sin(x)) = O(x^7)$
  - $O(e^n + an^{10}) = O(e^n)$
  - $O(n! + n^{50}) = O(n!)$
- An algorithm with an input of size n is said to be

  a). Linear if the running time is $O(n)$.

  b). Polynomial if the running time is $O(n^t)$ for some constant t.

  c). Exponential if the running time is $O(t^{h(n)})$ where t is some constant and $h(n)$ is a polynomial in n.

- Generally a problem that can be solved in polynomial time is considered feasible whereas anything worse than polynomial time is considered computationally infeasible - especially exponential time.
- N.B. If n is small enough, even complex algorithms become feasible.
- For example, most discussions on cryptanalysis of RSA centre on the task of factoring n into its 2 prime factors. The best known methods for doing this have a time complexity of:

$$f(n) = e^{\sqrt{ln(n) \cdot ln(ln(n))}} \tag{10}$$

- A graph of this equation is shown in figure 4.
- The "big O" is exponential running time so this is considered computationally infeasible (with big n, i.e. $n > 150$) and hence RSA is considered secure.

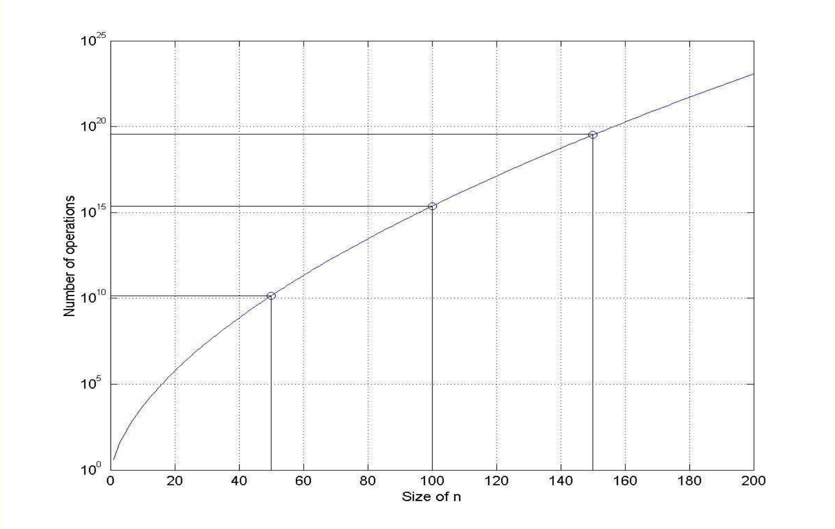


Figure 4: The number of operations required to factor an integer of size n.