Importance of Number Theory in Cryptography

[1]Dr.Ramesh.k, [2]Rajeshwari Patil

[1]Associate professor, Akkamahadevi Women's university, Vijayapur.
[2]Faculty, Akkamahadevi Women's university P.G.Center sindhanur.
[1]Department of computer science, [2]Department of mathematics, sindhanur(India)

Abstract: **Number theory is or in older usage arithmetic is a branch of pure mathematics devoted primarily to the study of the integer's .it is sometimes called the queen of mathematics because of its foundational place in the discipline. The number theory studies the prime numbers as well properties of objects made out of integers and also study to the matrices, determinant, cryptography. The cryptography is based on some specific areas of mathematics including number theory, linear algebra, algebra structures and we need to understand how cryptography is the study of mathematical techniques related to aspects of information security.**

Keywords: **Binary operations, Euclidean algorithm, Matrices, determinant, cryptography, RSA.**

I. Introduction

Number theory may be one of the "purest" branches of mathematics, but it has turned out to be one of the most useful when it comes to computer security. And also useful to the counting the number. The paper aims to introduce the reader to application of number theory in cryptography we will briefly talk about RSA keys in cryptography. Many tools like integer division, Euclidean algorithm, GCD, modulus etc.

1.1Binary Operations: In cryptography, we are interested in three binary operations applied to the set of integers. A binary operation takes two inputs and creates one output. Three common binary operations defined for integers are *addition, subtraction,* and *multiplication.* Each of these operations takes two inputs (*a* and *b*) and creates one output (*c*) as shown in Figure 1.1. The two inputs come from the set of integers; the output goes into the set of integers.
Note that *division* does not fit in this category because, as we will see shortly, it produces two outputs instead of one.

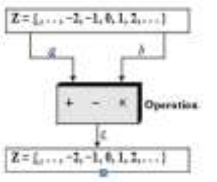

Z = {..., -2, -1, 0, 1, 2, ...}
a
b
+ - ×
Operation
c
Z = {..., -2, -1, 0, 1, 2, ...}

Figure 1.1 Three binary operation for the set of integers

1.1.1**Integer Division:** An integer arithmetic, if we divide *a* by *n*, we can get *q* and *r*. The relationship between these four integers can be shown as

$$\boxed{a = q \times n + r}$$

In this relation, *a* is called the *dividend; q,* the *quotient; n,* the *divisor;* and *r*, the *remainder*. Note that this is not an operation, because the result of dividing *a* by *n* is two integers, *q* and *r*. We can call it *division relation.*
Example 1.1: Assume that *a* = 255 and *n* =11. We can find *q* =23 and *r*= 2 using the division algorithm we have learned in arithmetic as shown in Figure 1.2.

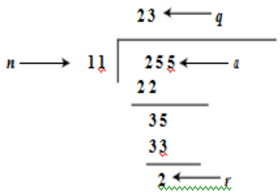


Figure 1.2 finding the quotient and the remainder

Most computer languages can find the quotient and the remainder using language-specific operators. For example, in the C language, the operator / can find the quotient and the operator % can find the remainder.

1.1.2 Two Restrictions*:* When we use the above division relationship in cryptography, we impose two restrictions. First, we require that the divisor be a positive integer (*n* =0). Second, we require that the remainder be a nonnegative integer (*r* ≥ 0). Figure 1.3 shows this relationship with the two above-mentioned restrictions.

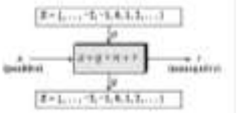


Figure 1.3: Two restriction of variable

Example 1.2: When we use a computer or a calculator, *r* and *q* are negative when *a* is negative. How can we apply the restriction that *r* needs to be positive? The solution is simple, we decrement the value of *q* by 1 and we add the value of *n* to *r* to make it positive.

$$-255 = (\mathbf{-23} \times 11) + (-\mathbf{2}) \leftrightarrow -255 = (\mathbf{-24} \times 11) + \mathbf{9}$$

We have decremented −23 to become −24 and added 11 to −2 to make it 9. The above relation is still valid.

1.1.3 The Graph of the Relation: We can show the above relation with the two restrictions on *n* and *r* using two graphs in Figure 1.4 The first one shows the case when *a* is positive; the second when *a* is negative.

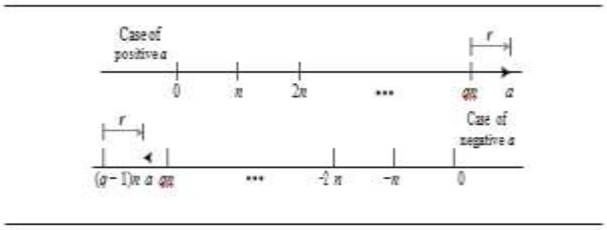


Figure 1.4 Graph of division algorithm

Starting from zero, the graph shows how we can reach the point representing the integer *a* on the line. In case of a positive *a*, we need to move *q* × *n* units to the right and then move extra *r* units in the same direction. In case of a negative *a*, we need to move (*q* − 1) × *n* units to the left (*q* is negative in this case) and then move *r* units in the opposite direction. In both cases the value of *r* is positive.

1.1.4 Divisibility: Let us briefly discuss divisibility, a topic we often encounter in cryptography. If *a* is not zero and we let *r* = 0 in the division relation, we get

a=q x n

We then say that *n* divides *a* (or *n* is a divisor of *a*). We can also say that *a* is divisible by *n.* When we are not interested in the value of *q*, we can write the above relation-ship as ***a*| *n*.** If the remainder is not zero, then *n* does not divide *a* and we can write the relationship as a=q x n.

Properties: Following are several properties of divisibility.

Property 1: if *a*| 1, then *a* = ±1.

Property 2: if *a*| *b* and *b*| *a*, then *a* = ±*b*.

Property 3: if *a*| *b* and *b*| *c*, then *a*| *c*.

Property 4: if *a*| *b* and *a*| *c*, then *a*| (*m* × *b* + *n* × *c*), where *m* and *n* are arbitrary integers.

1.1.5 Greatest Common Divisor: One integer often needed in cryptography is the greatest common divisor of two positive integers. Two positive integers may have many common divisors, but only one greatest common divisor. For example, the common divisors of 12 and 140 are 1, 2, and 4. However, the greatest common divisor is 4. See Figure 1.5.

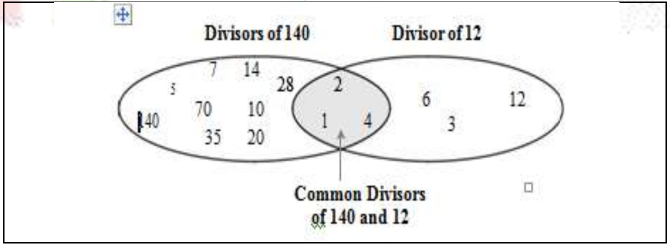
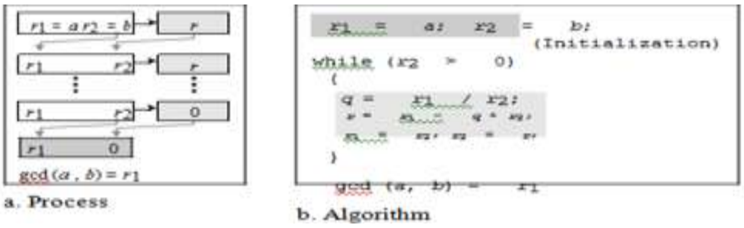


Figure 1.5 : common divisors of two integers

The greatest common divisor of two positive integer is the largest integer that can divide both integer**.**

1.1.6 Euclidean Algorithm: Finding the greatest common divisor (gcd) of two positive integers by listing all common divisors is not practical when the two integers are large. Fortunately, more than 2000 years ago a mathematician named Euclid developed an algorithm that can find the greatest common divisor of two positive integers.

Example 1.5: gcd (36, 10) =2, gcd (10, 6) =2, and so on. This means that instead of calculating gcd (36, 10), we can find gcd (2, 0). Figure 1.6shows how we use the above two facts to calculate gcd (*a*, *b*).



```
r1 = a;    r2 = b;    (Initialization)
while (r2 > 0)
  {
    q = r1 / r2;
    r = r1 - q * r2;
    r1 = r2;  r2 = r;
  }
gcd (a, b) = r1
```

Figure1.6: Euclidean algorithm

We use two variables, r_1 and r_2, to hold the changing values during the process of reduction. They are initialized to *a* and *b*. In each step, we calculate the remainder of r_1 divided by r_2 and store the result in the variable *r*. We then replace r_1 by r_2 and r_2 by *r*. The steps are continued until r_2 becomes 0. At this moment, we stop. The gcd (*a*, *b*) is r_1

When gcd (*a*, *b*) = 1, we say that *a* and *b* are relatively prime.

1.1.7 The Extended Euclidean Algorithm: Given two integers *a* and *b*, we often need to find other two integers, *s* and *t*, such that $s \times a + t \times b = \gcd(a, b)$

The extended Euclidean algorithm can calculate the gcd (*a*, *b*) and at the same time calculate the value of *s* and *t*. The algorithm and the process is shown in Figure 1.7.

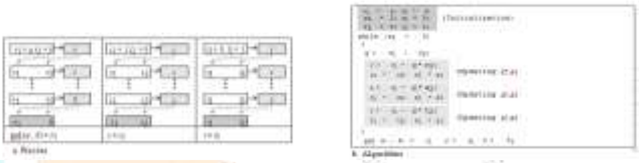


Figure 1.7: Extended Euclidean algorithm

In each step, r_1, r_2, and *r* have the same values in the Euclidean algorithm. The variables r_1 and r_2 are initialized to the values of *a* and *b*, respectively. The variables s_1 and s_2 are initial-ized to 1 and 0, respectively. The variables t_1 and t_2 are initialized to 0 and 1, respectively. The calculations of *r*, *s*, and *t* are similar, with one warning. Although *r* is the remainder of dividing r_1 by r_2, there is no such relationship between the other two sets. There is only one quotient, *q*, which is calculated as $r_1| r_2$ and used for the other two calculations.

1.2 Linear Diophantine Equations: Although we will see a very important application of the extended Euclidean algorithm in the next section, one immediate application is to find the solutions to the linear Diophantine equations of two variables, an equation of type $ax + by = c$. We need to find integer values for *x* and *y* that satisfy the equation. This type of equation has either no solution or an infinite number of solutions. Let $d = \gcd(a, b)$. If *d c*, then the equation has no solution. If $d \mid c$, then we have an infinite number of solutions. One of them is called the particular; the rest, general.

A linear Diophantine equation of two variables is $ax + by = c$.

1.3 Particular Solution:If d| c, a particular solution to the above equation can be found using the following steps

1.Reduce the equation to $a_1x + b_1y = c_1$ by dividing both sides of the equation by *d*. This is possible because *d* divides *a*, *b*, and *c* by the assumption.

2.Solve for *s* and *t* in the relation $a_1s + b_1t = 1$ using the extended Euclidean algorithm.

3.The particular solution can be found:

- **Particular solution:** $x_0 = (c/d)s$ and $y_0 =(c/d)t$.
- **General Solutions:** After finding the particular solution, the general solutions can be found:

General solutions: $x = x_0 + k\,(b/d)$ and $y = y_0 - k\,(a/d)$ where $k = 0, 1, 2, \ldots$

1.4 Modular Arithmetic:

The division relationship ($a = q \times n + r$) discussed in the previous section has two inputs (*a* and *n*) and two outputs (*q* and *r*). In modular arithmetic, we are interested in only one of the outputs, the remainder *r*. We don't care about the quotient *q*. In other words, we want to know what is the value of *r* when we divide *a* by *n*. This implies that we can change the above relation into a binary operator with two inputs *a* and *n* and one output *r*.

Modulo Operator: The above-mentioned binary operator is called the modulo operator and is shown as *mod*. The second input (*n*) is called the modulus**.** The output *r* is called the residue. Figure 1.8 shows the division relation compared with the modulo operator.

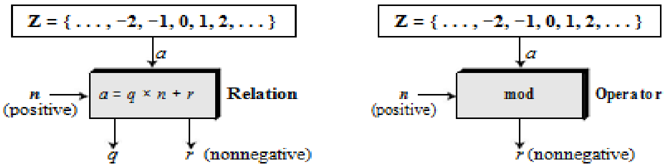


Figure 1.8 Division relation and modular operator

As Figure 1.8 shows, the modulo operator (mod) takes an integer (*a*) from the set Z and a positive modulus (*n*). The operator creates a nonnegative residue (*r*). We can say **a mod n=r**

1.5 Congruence: In cryptography, we often used the concept of congruence instead of equality. Map-ping from Z to Z_n is not one-to-one. Infinite members of **Z** can map to one member of $\mathbf{Z}_n$. For example, the result of 2 mod 10 = 2, 12 mod 10 = 2, 22 mod 2 = 2, and so on. In modular arithmetic, integers like 2, 12, and 22 are called congruent mod 10. To show that two integers are congruent, we use the congruence operator (≡). We add the phrase (mod *n*) to the right side of the congruence to define the value of modulus that makes the relationship valid.

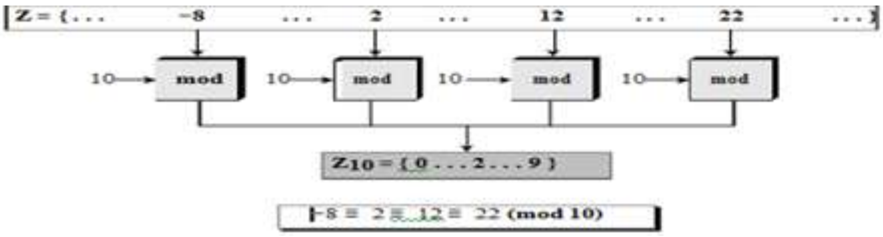

Z = { . . . −8 . . . 2 . . . 12 . . . 22 . . . }
10
mod
10
mod
10
mod
10
mod
Z10 = { 0 . . . 2 . . . 9 }
−8 ≡ 2 ≡ 12 ≡ 22 (mod 10)

Figure 1.9:congruence relationship

b.The phrase (mod *n*) that we insert at the right-hand side of the congruence opera-tor is just an indication of the destination set ($\mathbf{Z}_n$). We need to add this phrase to show what modulus is used in the mapping. The symbol *mod* used here does not have the same meaning as the binary operator. In other words, the symbol *mod* in 12 mod 10 is an operator; the phrase (mod 10) in 2 ≡ 12 (mod 10) means that the destination set is $\mathbf{Z}_{10}$.

1.6 Inverses: When we are working in modular arithmetic, we often need to find the inverse of a number relative to an operation. We are normally looking for an additive inverse (rela-tive to an addition operation) or a multiplicative inverse (relative to a multiplication operation).

1.6.1 Additive Inverse: In Z_n, two numbers *a* and *b* are additive inverses of each other if $a + b \equiv 0 \pmod{n}$

In $\mathbf{Z}_n$, the additive inverse of *a* can be calculated as $b = n - a$. For example, the additive inverse of 4 in $\mathbf{Z}_{10}$ is 10 − 4 = 6.

- In modular arithmetic, each integer has an additive inverse.
- The sum of an integer and its additive inverse is congruent to 0 modulo *n*.

Note that in modular arithmetic, each number has an additive inverse and the inverse is unique; each number has one and only one additive inverse. However, the inverse of the number may be the number itself.

1.6.2 Multiplicative Inverse:

In $\mathbf{Z}_n$, two numbers *a* and *b* are the multiplicative inverse of each other if

$$a \times b \equiv 1 \pmod{n}$$

- In modular arithmetic, an integer may or may not have a multiplicative inverse.
- When it does, the product of the integer and its multiplicative inverse is congruent to 1 modulo n.

It can be proved that *a* has a multiplicative inverse in Z_n if and only if gcd (*n*, *a*) = 1.In this case, *a* and *n* are said to be relatively prime.

The integer *a* in Z_n has a multiplicative inverse if and only if gcd $(n, a) \equiv 1 \pmod{n}$

II. MATRICES

A matrix is a rectangular array of $l \times m$ elements, in which *l* is the number of rows and *m* is the number of columns. A matrix is normally denoted with a boldface uppercase letter such as **A**. The element a_{ij} is located in the *i*th row and *j*th column. Although the elements can be a set of numbers, we discuss only matrices with elements in Z. Figure 2.1 shows a matrix.

If a matrix has only one row (*l* = 1), it is called a row matrix**;** if it has only one col-umn (*m* = 1), it is called a column matrix**.** In a square matrix, in which there is the matrix A

$$\textbf{Matrix A:}\quad \begin{array}{cc} & m\text{ columns} \\ l\text{ rows} & \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{bmatrix} \end{array}$$

Figure 2.1:A matrix of size 1x m

Same number of rows and columns (1=m), the elements $a_{11,}a_{22,}\ldots\ldots a_{mm}$ make the main diagonal .an additive identity matrix ,denoted as 0,is a matrix with all rows and columns sets as 0,is a matrix with all rows and columns sets to 0’ s .an identity matrix ,denoted as I ,is a square matrix with 1s on the main diagonal and 0s .figure 1.19 shows some examples of matrices with elements from Z.

2.1 Operations and Relations

In linear algebra, one relation (equality) and four operations (addition, subtraction, multiplication, and scalar multiplication) are defined for matrices.

2.2 Equality:

Two matrices are equal if they have the same number of rows and columns and the corresponding elements are equal. In other words, A = B if we have $a_{ij} = b_{ij}$ for all *i*'s and *j*'s.

2.3 Addition and Subtraction

Two matrices can be added if they have the same number of columns and rows. This addition is shown as C = A + B. In this case, the resulting matrix C has also the same number of rows and columns as A or B. Each element of C is the sum of the two corresponding elements of A and B: $c_{ij} = a_{ij} + b_{ij}$. Subtraction is the same except that each element of B is subtracted from the corresponding element of **A**: $d_{ij} = a_{ij} - b_{ij}$

2.4 Determinant: The determinant of a square matrix A of size $m \times m$ denoted as det (A) is a scalar calculated recursively as shown below:

1 .If $m = 1$, det (A) = a_{11}
2. If $m > 1$, det (A) $=(-1)^{i+j} \times a_{ij} \times \det(A_{ij})$ i=1,2,3,…..m

Where $\mathbf{A}_{ij}$ is a matrix obtained from **A** by deleting the *i*th row and *j*th column.

The determinant is defined only for a square matrix.

2.5 Inverses : Matrices have both additive and multiplicative inverses.

Additive Inverse: The additive inverse of matrix A is another matrix B such that A + B = 0. In other words, we have $b_{ij} = -a_{ij}$ for all values of *i* and *j*. Normally the additive inverse of A is defined by −A.

Multiplicative Inverse: The multiplicative inverse is defined only for square matrices. The multiplicative inverse of a square matrix A is a square matrix B such that A × B = B × A = I. Normally the multiplicative inverse of **A** is defined by A^{-1}. The multiplicative inverse exists only if the **(A)** has a multiplicative inverse in the corresponding set. Since no integer has a multiplicative inverse in Z, there is no multiplicative inverse of a matrix in Z. However, matrices with real elements have matrices only if det (A) ≠ 0.

Multiplicative inverses are only defined for square matrices

2.6 Matrices Cryptography uses residue matrices: matrices in all elements are in Z_n. All operations on residue matrices are performed the same as for the integer matrices except that the operations are done in modular arithmetic. One interesting result is that a residue matrix has a multiplicative inverse if the determinant of the matrix has a multiplicative inverse in Z_n. In other words, a residue matrix has a multiplicative inverse if gcd (det(**A**), *n*) = 1.

Example 1.2*:* Figure 2.2 shows a residue matrix A in $\mathbf{Z}_{26}$ and its multiplicative inverse A^{-1}. We have det(**A**) = 21 which has the multiplicative inverse 5 in Z_{26}. Note that when we multiply the two matrices, the result is the multiplicative identity matrix in Z_{26}.

$$\mathbf{A} = \begin{bmatrix} 3 & 5 & 7 & 2 \\ 1 & 4 & 7 & 2 \\ 6 & 3 & 9 & 17 \\ 13 & 5 & 4 & 16 \end{bmatrix} \qquad \mathbf{A}^{-1} = \begin{bmatrix} 15 & 21 & 0 & 15 \\ 23 & 9 & 0 & 22 \\ 15 & 16 & 18 & 3 \\ 24 & 7 & 15 & 3 \end{bmatrix}$$

$$\det(\mathbf{A}) = 21 \qquad \det(\mathbf{A}^{-1}) = 5$$

Figure2.2 Residue matrix and it's multiplicative inverse

Congruence*:* Two matrices are congruent modulo *n*, written as **A** ≡ **B** (mod *n)*, if they have the same number of rows and columns and all corresponding elements are congruent modulo *n*. In other words, **A** ≡ **B** (mod *n*) if $a_{ij} \equiv b_{ij}$ (mod *n*) for all *i*'s and *j*'s.

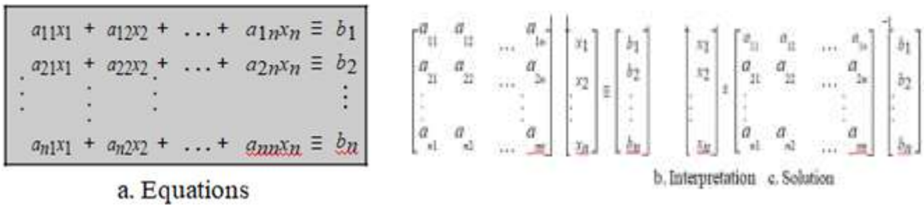
Linear Congruence: Cryptography often involves solving an equation or a set of equations of one or more variables with coefficient in $\mathbf{Z}_n$. This section shows how to solve equations when the power of each variable is 1 (linear equation).

Single-Variable Linear Equations: Let us see how we can solve equations involving a single variable that is, equations of the form $ax \equiv b$ (mod *n*). An equation of this type might have no solution or a limited number of solutions. Assume that the gcd (*a*, *n*) = *d*. If *d b*, there is no solution. If *d*| *b*, there are *d* solutions.

If *d*| *b*, we use the following strategy to find the solutions:

- Reduce the equation by dividing both sides of the equation (including the modulus) by *d*.
- Multiply both sides by the multiplicative inverse of *a*| gcd (*a*, *n*) to find the particular solution x_0.
- The general solutions are $x = x_0 + k$ (*n*| *d*) for *k* = 0, 1, . . . , (*d* − 1).

2.7 Set of Linear Equations: We can also solve a set of linear equations with the same modulus if the matrix formed from the coefficients of the variables is invertible. We make three matrices. The first is the square matrix made from the coefficients of variables. The second is a column matrix made from the variables. The third is a column matrix made from the values at the right-hand side of the congruence operator. We can interpret the set of equations as matrix multiplication. If both sides of congruence are multiplied by the multiplicative inverse of the first matrix, the result is the variable matrix at the right-hand side, which means the problem can be solved by a matrix multiplication as shown in Figure 2.3



a. Equations b. Interpretation c. Solution

Figure 2.3: Set of linear equation

III. CRYPTOGRAPHY

3.1 A cryptosystem is a five-tuple (M,C,K,E,D),
where the following condition are satisfied
1. M is a finite set of possible plaintexts or messages
2. C is a finite set of possible ciphertexts or cryptograms
3. K is a finite set of possible keys
4. For each K ∈K, there is an encryption rule EK ∈E and a corresponding decryption rule DK ∈ D. Each EK : M → C and DK : C → M are functions such that DK(EK(x)) = x for every message x∈M.

The RSA Cryptosystem The various observations just stated form the basis for the RSA public-key cryptosystem, which was invented at MIT in 1977 by Ronald Rivest, Adi Shamir and Leonard Adleman.

The public key in this cryptosystem consists of the value n, which is called the modulus, and the value e, which is called the public exponent. The private key consists of the modulus n and the value d, which is called the private exponent.

3.2. An RSA public-key / private-key pair can be generated by the following steps:

1. Generate a pair of large, random primes p and q. 2. Compute the modulus n as n = pq. 3. Select an odd public exponent e between 3 and n-1 that is relatively prime to p-1 and q-1. 4. Compute the private exponent d from e, p and q. (See below.) 5. Output (n, e) as the public key and (n, d) as the private key.

The encryption operation in the RSA cryptosystem is exponentiation to the eth power modulo n.

$$c = \mathrm{ENCRYPT}\,(m) = m^e \bmod n\ .$$

The input m is the message; the output c is the resulting ciphertext. In practice, the message m is typically some kind of appropriately formatted key to be shared. The actual message is encrypted with the shared key using a traditional encryption algorithm. This construction makes it possible to encrypt a message of any length with only one exponentiation.

The decryption operation is exponentiation to the dth power modulo n:

$$m = \mathrm{DECRYPT}\,(c) = c^d \bmod n\ .$$

The relationship between the exponents e and d ensures that encryption and decryption are inverses, so that the decryption operation recovers the original message m. Without the private key (n, d) (or equivalently the prime factors p and q), it's difficult to recover m from c. Consequently, n and e can be made public without compromising security, which is the basic requirement for a public-key cryptosystem.

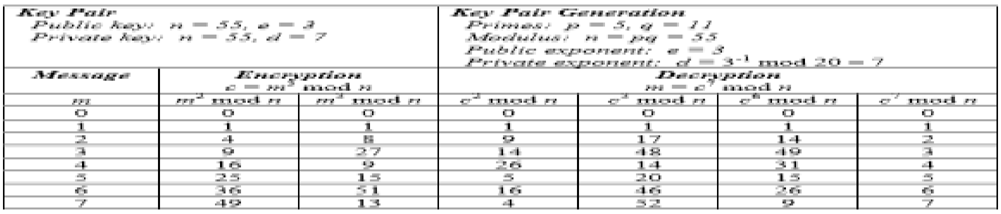
The fact that the encryption and decryption operations are inverses and operate on the same set of inputs also means that the operations can be employed in reverse order to obtain a digital signature scheme following Diffie and Hellman's model. A message can be digitally signed by applying the decryption operation to it, i.e., by exponentiating it to the dth power:

$$s = \mathrm{SIGN}\,(m) = m^d \bmod n\ .$$

The digital signature can then be verified by applying the encryption operation to it and comparing the result with and/or recovering the message:

$$m = \mathrm{VERIFY}\,(s) = s^e \bmod n\ .$$

In practice, the plaintext m is generally some function of the message, for instance a formatted one-way hash of the message. This makes it possible to sign a message of any length with only one exponentiation. Figure 1.1:gives a small example showing the encryption of values m from 0 to 9 as well as decryptions of the resulting cipher texts. The exponentiation is optimized as suggested above. To compute m^3 mod n, one first computes m^2 mod n with one modular squaring, then m^3 mod n with a modular multiplication by m. The decryption is done similarly: One first computes c^2 mod n, then c^2 mod n, c^6 mod n, and c^7 mod n by alternating modular squaring and modular multiplication.



Key Pair Public key: $n = 55, e = 3$ Private key: $n = 55, d = 7$			Key Pair Generation Primes: $p = 5, q = 11$ Modulus: $n = pq = 55$ Public exponent: $e = 3$ Private exponent: $d = 3^{-1}$ mod $20 = 7$			
Message	Encryption $c = m^3$ mod n		Decryption $m = c^7$ mod n			
m	m^2 mod n	m^3 mod n	c^2 mod n	c^3 mod n	c^6 mod n	c^7 mod n
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	4	8	9	17	14	2
3	9	27	14	48	49	3
4	16	9	26	14	31	4
5	25	15	5	20	15	5
6	36	51	16	46	26	6
7	49	13	4	52	9	7

Figure 3.2: Public key cryptography

3.3. The mathematics of the RSA public-key cryptosystem:

Cryptosystem is used keys and sign messages. This requires a careful analysis of various methods for relating of the value m to the actual message or key, some of which are much better for security than others. We have needed to understand the impact of various proposed improvements to integer factorization methods, especially in terms of recommended key sizes. This requires assessment of the effectiveness of those methods.

Because of the widespread deployment of the RSA algorithm, many other researchers are looking at these same problems today. We benefit significantly from their work as we look to improve our own products and provide guidance to our customers. Our work in industry standards aims to promote the adoption of the best practices we have learned. The Work Has Just Begun Simple concepts in mathematics – prime numbers, integer factorization, modular exponentiation – have had a dramatic impact on computer security, particularly for online commerce. The theory is working well in practice through algorithms like Diffie Hellman key agreement, the RSA public-key cryptosystem and more recently elliptic curve cryptography.

In cryptography, “it’s not broken” is no reason to avoid trying to fix it. Mathematicians still don’t know whether or not there are faster methods for integer factorization than the ones currently available. Research is needed to try to find faster methods, as well to try to prove that there aren’t any. A related research problem is to confirm whether modular root-extraction is or isn’t as hard as integer factorization.

Interestingly, much faster methods for integer factorization already exist in theory, but they run on computers that haven’t yet been built. In particular, if one could build a full scale quantum computer, it will be possible to break a large number into its factors essentially as easily as it is to put the number together by multiplication. (Such a computer would also break the Diffie-Hellman and elliptic curve algorithms.)

In case one or more of the current public-key cryptosystems is broken in the future, it would be helpful to have alternatives to choose from. This is another important area for research. What other hard problems in mathematics are there from which a public-key cryptosystem and digital signature scheme might be derived?

Mathematics has many more applications in computer security than just public-key cryptography, of course. The design and analysis of more traditional encryption algorithms and one-way function also has a strong mathematical component, although perhaps not one so elegant as for public-key cryptography. Intriguing mathematical constructions have also led to new types of cryptography – like identity-based encryption, a form of public-key cryptography where one’s name or e-mail address itself becomes the public key, avoiding the need for a directory. More groundbreaking applications are likely to emerge over time as knowledgeable people continue to search what’s concealed within mathematics. Who knows what other useful things we might find by exploring an otherwise obscure formula?

Computing the Private Exponent Let n be the product of two distinct prime numbers p and q, and let e be the public exponent as defined above. Let L = LCM (p-1, q-1) denote the least common multiple of p-1 and q-1. The private exponent d for the RSA cryptosystem is any integer solution to the congruence

$$de \equiv 1 \text{ mod } L.$$

The value d is the inverse of e modulo L. The requirement that e be relatively prime to p1 and q-1 ensures that an The RSA cryptosystem works because exponentiation to the d^{th} power modulo n is the inverse of exponentiation to the e^{th} power when the exponents d and e are inverses modulo L. That is, for all m between 0 and n-1, inverse exists. Modular inverses are easy to find with the Extended Euclidean Algorithm or similar methods.

$$m \equiv (m^e)^d \bmod n \quad .$$

The proof of this fact is left as an exercise to the reader. This result follows the Chinese remainder theorem.

IV. CONCLUSION

In this paper perceive that every number theory plays an important role in cryptography to hide information .many tools like primes, divisors, congruence, two restriction variables, Euclidean algorithm, gcd, matrices etc plays important role in cryptography, the cryptography used for security purpose. This gives an idea of cryptosystem in the context of algebra and number theory. The abstract algebra and number theory which is particularly useful for public key cryptosystem as well as cryptographic techniques described in this paper.

References

[1] N. Koblitz. *"Algebraic aspects of cryptography"*. Springer-Verlag, 1998.

[2] M.D. Larsen. *"Introduction to modern algebraic concepts"*. Addison-WesleyPublishing Company, 1969.

[3] J.L. Massey and J.K. Omura. "A new multiplicative algorithm over finite fieldsand its applicability in public key cryptography". *Presented at EUROCRYPT'83 Udine, Italy*, 1983

[4] T. Nagell. *"Introduction to number theory"*. Chelsea Publishing Company,1981. [2] Neal Koblitz. Elliptic curve cryptosystems. Math. Comp., 48(177):203{209,1987.

[5] William J. LeVeque. Fundamentals of Number Theory. Dover Publications,Inc., New York, 1977.

[6] Victor Shoup. A Computational Introduction to Number Theory and Algebra.CambridgUniversityPress,2008.

[7] Joseph H. Silverman. An Introduction to the Theory of Elliptic Curves, 2006.Slides of a summer school on Computational Number Theory and Applicationsto Cryptography at University of Wyoming.
http://www.math.brown.edu/ _ jhs/Presentations/WyomingEllipticCurve.pdf.

[8] Joseph H. Silverman and John Tate. Rational Points on Elliptic Curves.Springer-Verlag, New York, 1992.

[9] Harold M. Stark. An Introduction to Number Theory. The MIT Press, Cam-bridge, Massachusetts, 1970.

[10] Wade Trappe and Lawrence C. Washington. Introduction to Cryptography withCoding Theory. Pearson Prentice Hall, Upper Saddle River, New Jersey 07458,2006.

[11] Lawrence C. Washington. Elliptic Curves: Number Theory and Cryptography.Chapman & Hall/CRC, Boca Raton, 2003.

[12] Eric W. Weisstein. Rabin-Miller Strong Pseudoprime Test. From Math-World { A Wolfram Web Resource. http://mathworld.wolfram.com/Rabin-}