CHAPTER 3

OBJECTIVE OF THE PRESENT WORK

3.1 Introduction

Confusion is similarity between two entities, which make then look alike. Diffusion is, though the object is visible in contour, it cannot be deciphered as a specific thing due to diffusivity of the object. Claude Shannon [SHAN48], [SHAN49] of MIT was the first to formulate the main problems of the Science of Information. He formulated the principles of Confusion and Diffusion in Information Theory in 1946. In fact, his principles of Confusion and Diffusion are taken as the most important concepts of information theory.

C. Jean et al have shown the first deterministic polynomial-time algorithm that factors an RSA modulus $N$ given the pair of public and secret exponents $e$ and $d$, provided that $e \cdot d < N^2$. The algorithm is a variant of Coppersmith’s technique for finding small roots of univariate modular polynomial equations. They have also provided a generalization to the case of unbalanced prime factors. But the problem of the deterministic polynomial-time equivalence between finding $d$ and factoring $N$ is not entirely solved, because finding an algorithm for $e \cdot d > N^2$ remains an open problem.
3.2 Objective of this work

Objective of this research is to –

i) Build an Encryption and Decryption process - a stream cipher using natural numbers. Generate a group of invertible elements from any natural number and then show how this group can be used with Claude Shannon’s principles for generation of a simple (yet as secure, as the one that is generated with the help of larger primes) encryption and decryption (ED) process. The proposed ED Process is a stream ciphering uses simple modular arithmetic to generate the pseudorandom blocks of invertible elements as the stream of keys.

ii) Encode a Key in Hierarchical Single Key-Lock (SKL) mechanism based on Chinese Remainder Theorem (CRT) using Gaussian Farey fractions. Numerator and denominators of Gaussian Farey fractions are the natural numbers. Here key number is encoded using CRT, Tribes of Gaussian or Rational Farey Numbers are used to encode the key number. The advantages of this encoding are two fold, it is not required to decode the key to determine the access rights and it is also not required to derive the key number for already existing files. This method increases the security level of the system by encoding the key using arbitrary Farey Number which acts a thin layer encryption.

The discussion in the following sections in this chapter gives the basic reason to go for natural numbers for encryption and decryption process or encoding access control mechanism key.
3.3 Encryption and Decryption Using Natural Numbers

3.3.1 Limitations of prime numbers in Cryptography

The security of RSA is primarily tied to the difficulty of factoring large composite numbers. The first step in configuring an RSA system is to choose two large prime numbers. The next step is to multiply the primes together to create a composite number whose factors are the two original primes.

As it turns out, generating two primes of adequate size is not difficult, nor is multiplying them together. The problem lies in factoring the resulting composite number given two sufficiently large prime numbers; it is believed to be infeasible to factor the product into its components in a finite amount of time. But over time, because of a gradual increase in computer power, the definition of "sufficiently large" has changed, and the RSA system adapts by increasing the recommended number of binary bits in the prime numbers. At the time of writing, the two primes are regarded as sufficiently secure if they are 2048 bits long. The product of two such primes is twice as long — 4096 bits, or about 1234 decimal digits.

Much of modern banking, commerce and diplomacy depend on the security of the RSA system or an equivalent. Confidence in these systems hinges on the assumption that large composite numbers cannot be factored in any reasonable time. Unfortunately, this is an assumption, it hasn't been
proven, and over time we see improvements in the efficiency of factoring algorithms.

Those responsible for computer security sometimes organize contests in which people are encouraged to think of new, more ingenious ways to factor composites. The RSA Factoring Challenge is one such contest, in which monetary rewards are offered for the successful factorization of "RSA Numbers", the keys to the RSA system.

In 2005, a 640-bit RSA number eponymous named RSA-640 was successfully factored by a German team, an effort that required five months and 80 processors. From this and similar results we can conclude that our trust in 4096-bit RSA numbers is well-placed ... for now.

**Distribution of primes**

The prime number theorem states that the primes near $n$ are spaced on an average one every $\ln (n)$ integers. Thus, on average, one would have to test on the order of $\ln (n)$ integers before a prime is found. Excluding even integers and integers ending with 5, the correct figure is $0.4 \ln (n)$. For example, if a prime on the order of magnitude of $2^{200}$ were sought, then about 55 trials would be needed to find a prime. However, this figure is just an average. In some places along the number line, primes are closely packed, and in other places there are large gaps.
3.3.2 Prime factorization

If it becomes practical, a new kind of computer may erode this confidence. Some preliminary examples of a "quantum computer" have been tested in laboratories and, although there is still uncertainly that such a scheme can be made reliable; if it succeeds it bodes ill for RSA.

A conventional computer attacks a problem in a series of steps. For example a simple integer factoring program might use this approach to factoring a test number:

Sophisticated factoring programs use various strategies to improve on this simple algorithm, but basically, as far as is known at the time of writing, one is reduced to laboriously dividing a test number by many, many possible factors, and as the test number grows larger, the problem's severity increases beyond polynomial time. In everyday terms, in a world of conventional computers, schemes like RSA are quite secure.

Factoring problem

Factoring is the act of splitting an integer into a set of smaller integers (factors) which, when multiplied together, form the original integer. For example, the factors of 15 are 3 and 5; the factoring problem is to find 3 and 5 when given 15. Prime factorization requires splitting an integer into factors that are prime numbers; every integer has a unique prime factorization. Multiplying two prime integers together is easy, but as far as we know, factoring the product of two (or more) prime numbers is much more difficult.
Algorithm 3.1: Conventional Method of Prime Factorization

**Input:** Natural numbers \( n > 1 \)  
**Output:** \( n \) if \( n \) is Prime / Factors of \( n \)  
**Method:** Conventional method  
**Steps:**  
1. Start with the first prime number 2.  
2. Divide \( n \) by the prime. If remainder = 0, replace \( n \) with the result and add the divisor to the list of factors  
3. Repeat step 2 until it fails.  
4. Choose the next prime number.  
5. if new prime \( \geq \sqrt{n} \) goto step 6  
   else goto step 2.  
6. Return the list of factors. If the only factor is the test number, the number is prime

Factoring is the underlying, presumably hard problem upon which several public-key cryptosystems are based, including the RSA algorithm. Factoring an RSA modulus would allow an attacker to figure out the private key; thus, anyone who can factor the modulus can decrypt messages and forge signatures.

It is not necessarily true that a large number is more difficult to factor than a smaller number. For example, the number \( 10^{1000} \) is easy to factor, while the 155-digit number RSA-155 was factored after seven months of extensive computations. What is true in general is that a number
with large prime factors is more difficult to factor than a number with small prime factors (still, the running time of some factoring algorithms depends on the size of the number only and not on the size of its prime factors). This is why the size of the modulus in the RSA algorithm determines how secure an actual use of the RSA cryptosystem is. Namely, an RSA modulus is the product of two large primes; with a larger modulus, the primes become larger and hence an attacker needs more time to factor it. Yet, remember that a number with large prime factors might possess certain properties making it easy to factor. For example, this is the case if the prime factors are very close to each other.

**Best factoring methods in use today**

Factoring algorithms come in two flavors, special purpose and general purpose; the efficiency of the former depends on the unknown factors, whereas the efficiency of the latter depends on the number to be factored. Special-purpose algorithms are best for factoring numbers with small factors, but the numbers used for the modulus in the RSA cryptosystem do not have any small factors. Therefore, general-purpose factoring algorithms are the more important ones in the context of cryptographic systems and their security.

Special-purpose factoring algorithms include the Pollard rho method [POL75], with expected running time $O(\sqrt{p})$, and the Pollard $p - 1$ method [POL74], with running time $O(p')$, where $p'$ is the largest prime factor of $p - 1$. The Pollard $p + 1$ method is also a special purpose factoring algorithm, with running time $O(p')$, where $p'$ is the largest prime factor of $p + 1$. All of these take an amount of time that is exponential in the size (bit length) of
\( p \), the prime factor that they find; thus these algorithms are too slow for most factoring jobs. The elliptic curve method (ECM) \([\text{LEN87}]\) is superior to these; its asymptotic running time is \( O\left(e^{\sqrt{2(\ln p)(\ln \ln p)}}\right) \). The ECM is often used in practice to find factors of randomly generated numbers; it is not fast enough to factor a large modulus of the kind used in the RSA cryptosystem.

The best general-purpose factoring algorithm today is the Number Field Sieve (NFS) \([\text{BLP94}, \text{BLZ94}]\), which runs in time approximately \( O\left(e^{1.9(\ln n)^{2/3} (\ln \ln n)^{1/3}}\right) \). Previously, the most widely used general purpose algorithm was the Multiple Polynomial Quadratic Sieve (MPQS) \([\text{SIL87}]\), which has running time \( O\left(e^{(\ln \ln n)^{1/2} (\ln n)^{1/2}}\right) \).

**Improving the capability of factoring**

Factoring has become easier over the last 15 years for three reasons: computer hardware has become more powerful, computers have become more plentiful and inexpensive, and better factoring algorithms have emerged.

Recently, the number of computers has increased dramatically. While the computers have become steadily more powerful, the increase in their power has not compared to their increase in number. Since some factoring algorithms can be done with multiple computers working together, the more computers devoted to a problem, the faster the problem can be solved. Unlike the hardware improvement factor, prevalence of computers does not make the RSA cryptosystem more secure.
Better factoring algorithms have been more help to the attacker. RSA cryptosystem and cryptography in general have attracted much attention, so has the factoring problem and many researchers have found new factoring methods or improved upon others. This has made factoring easier for numbers of any size, irrespective of the speed of the hardware.

Increasing the key size can offset any decrease in security due to factoring algorithm improvements. In fact, between general computer hardware improvements and special-purpose hardware improvements, increases in key size (maintaining a constant speed of RSA algorithm operations) have kept pace or exceeded increases in algorithm efficiency, resulting in no net loss of security. As long as hardware continues to improve at a faster rate than the rate at which the complexity of factoring algorithms decreases, the security of the RSA cryptosystem will increase, assuming users regularly increase their key sizes by appropriate amounts.

The open question is how much faster factoring algorithms can get; there could be some intrinsic limit to factoring speed, but this limit remains unknown. However, if an ‘‘easy’’ solution to the factoring problem can be found, the associated increase in key sizes will render the RSA system impractical.

Factoring is widely believed to be a hard problem, but this has not yet been proven. Therefore, there remains a possibility that an easy factoring algorithm will be discovered. This development, which could seriously weaken the RSA cryptosystem, would be highly surprising and the possibility is considered remote by the researchers most active in factoring research.
There is also the possibility someone will prove factoring is difficult. Such a development, while unexpected at the current state of theoretical factoring research, would guarantee the security of the RSA cryptosystem beyond a certain key size. Even if no breakthroughs are discovered in factoring algorithms, both factoring and discrete logarithm problems can be solved efficiently on a quantum computer if one is ever developed.

3.4 Study of Access Control Mechanism

Access control is an important issue in information security. It is a necessary mechanism for protecting data in a computer system. Resource sharing is one of the most important benefits in the client/server architecture or distributed environment. For example, a file server provides a centralized management and shares the storage of remote data among a number of workstations. If distributed workstations want to access some files, they send their requests to the file server via a communication network. The file server facilitates resource sharing among the autonomous workstations and has some obvious benefits, such as easy management, economy, and so on. On the other hand, a database is also a very important element in resource sharing in modern computer usage. It allows sharing of data among many users or applications. Now, almost every business application is supported by some sort of database in order to work, and thus more and more secret data are being stored in databases for easy access.

Part of this work is published in National Conference on Information & Communication Technology, 28th & 29th March, 2008, NHCE, Bangalore.
However, the resource sharing systems are dangerous if they are without information protection mechanisms. Because the amount of secret information, such as business plans, military maps, and so on, are available to illegitimate or unauthorized users. More and more threats to secret data stored in computer systems are increasing exponentially with the growth of network users and technological developments. Therefore, how to achieve privacy and integrity of information is becoming an important issue in modern computer security.

Access control is one of the efficient ways to prevent information from being destroyed, altered, disclosed, or copied by unauthorized users. Access control makes decisions on all users accessing a system. The policies of access control define the rules which permit certain subjects (such as users) to invoke objects (such as programs or files) in a computer system and which give them access right to operations (such as to executions, own, reading, or writing). Typically, the policies of access control can be defined according to five components [SAB05].

i. Users: The policy can identify what a user is allowed to do. The term “users” could refer to the people sitting at a terminal or workstation or the processes which run the computer system.

ii. Resources: The policy can specify what resources a user can access. The term “resources” could refer to the programs, services, and data accessed by users.

iii. Operations: The policy can specify what operations the user is permitted to invoke from the resource. The term “operations” refer
to the actions that can be performed on a resource. For instance, one user may be permitted to write a file, whereas another user may only read the file.

iv. Authority: The authority policy is to ensure that the access control is implemented according to the policies of the management of an organization. The term “authority” refers to the legitimate power to make policy decisions.

v. Domain: The policy can specify the boundary of the resources or users. For instance, a manager usually has higher authority than the resources and people in a department.

Authorization control is divided into three categories.

a) Discretionary access control (DAC).

The first category is called discretionary access control. In this category, the system leaves the specification of access control policies to individual users and controls the access of users by authenticating the identity after proper verification.

b) mandatory access control (MAC)

The second category is called mandatory access control. The subjects and objects in MAC are classified into many clearances and classifications. Furthermore, the access right for each subject is defined by a system administrator. Typically, the policies of MAC
must satisfy two restrictions, namely, (a) no read up and (b) no write down.

However, the DAC and MAC models are not flexible enough to provide a variety of access control policies. Since the access policies of these models are pre-defined and have been built into the access control mechanisms, they cannot support the dynamic requirements needed in modern application environments.

c) role-based access control (RBAC)

In order to deal with this problem, the third category of an access control model, called role-based access control, is presented. Currently, RBAC is the most popular access control model. RBAC can not only support the policies of DAC and MAC but can also support more complex policies. The roles of RBAC are created for various job functions in an organization, and users are assigned roles based on their responsibilities and qualifications. In this model, users can easily be reassigned from one role to another. This greatly simplifies the management of access policies [SAND99].

So far, many schemes in the three categories have been proposed for controlling access to computer systems [FER99], [HIT00], [SAND96], [SAND99]. In these schemes, once a user tries to access a protected file, the user needs to provide proof to the system that he/she is the authorized user. Password or key verification is the most popular technique for identifying an authorized user. Once an outside user wants to access certain resources
in a system, the user has to submit a secret password or key for verifying the access privilege to the resource.

However, some complex computer systems may need to change the access right with time. Thus, a file can sometimes be accessed and sometimes not. For example, a file may be updated every morning by the system administrator, and we wish that no one can access this file during this time period to avoid any data inconsistency. The property is very flexible and useful in many modern applications. Such an access control scheme is called time-constraint access control. In the time-constraint access control model, the access policies are associated with the time period. In different time periods, a user has different access rights.

Table 3.1: A typical access control matrix

<table>
<thead>
<tr>
<th>Users / Files</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$F_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>READ</td>
<td>OWN</td>
<td>READ</td>
<td>WRITE</td>
</tr>
<tr>
<td>$U_2$</td>
<td>WRITE</td>
<td>READ</td>
<td>NULL</td>
<td>OWN</td>
</tr>
<tr>
<td>$U_3$</td>
<td>READ</td>
<td>WRITE</td>
<td>OWN</td>
<td>NULL</td>
</tr>
</tbody>
</table>

So far, many authentication mechanisms have been proposed to authenticate a user’s access privileges. In 1972, Graham and Denning [GRAN72] first introduced an abstract protection model. In their model, the security state of an access control system is defined by an access control matrix, where rows correspond to users, columns correspond to the protected files, and the entries correspond to the lists of users with access rights to the protected files. A simple access control matrix for a file system is shown in Table 3.1. The entry $a_{ij}$ stands for the access right of the user $U_i$ with respect to the file $F_j$. If one user has no access right to a file, the corresponding entity $a_{ij}$ in the matrix is assigned a “null” value.
For example, the user $U_1$ has access rights to read the files $F_1$ and $F_3$ and to write the file $F_4$. When a user tries to access a file, the system will check the matrix to verify the user’s access right. The access control model is simple and easy to implement.

In general, however, there is a huge amount of data stored in a file system. Therefore, the size of the access control matrix would be very large, and most of the entities in the matrix are nulls because each user is usually only allowed access to a subject of the files. If we store the whole access control matrix, the memory needed for the whole matrix becomes impractically large, and the utilization of storage is very low. On the other hand, the system must maintain and protect the matrix from being modified by an intruder.

To overcome the problems faced by the above scheme, Wu and Hwang [WU84] proposed a single-key-lock (SKL) model for implementing the access control system as shown in the figure 3.1. In their model, each user $U_i$ is associated with a key vector $K_i$, and each file $F_j$ is associated with a lock vector $L_j$.

Therefore, the access right $a_{ij}$ for $U_i$ to $F_j$ are be formulated as

$$a_{ij} = f(K_i, L_j),$$

where $f()$ is a predefined function.

Since then, several methods have been developed for a single-key-lock access control model. Chang [CHNG86] [CHNG87] [CHNG97] proposed three of them based on the Chinese remainder theorem, Euler’s theorem, and prime factorization. Laih et al. [HARN89] proposed a single-
key-lock model based on Newton’s interpolating polynomials. The previous schemes work well, but they are impractical for complex computer systems.

![Diagram of Key-Lock Model](image)

**Figure 3.1: Single Key Lock Pair Access Control Mechanism System**

A drawback to this kind of model is that if the user has access to the access control system, the user can modify the key such that he/she has access to unauthorized resource. However the key can be encoded with available encryption techniques and stored. However the key needs to be decoded before the access rights are to be verified. The proposed method of encoding of the access control key using tribes of Farey numbers does not require decoding of the encoded key, the access rights can be calculated directly using properties of tribes Farey numbers. The encryption is a thing layer visible only while encoding.
3.5 Encoding Hierarchical Access Control Mechanism key using Tribes of Gaussian Farey Fractions

This thesis proposes a method to encode a key in Hierarchical Single Key-Lock (SKL) pair mechanism based on Chinese remainder theorem (CRT). Tribes of Gaussian or Rational Farey Fractions are used to encode the key. Kim S. Lee et al. [LEE92] in their Hierarchical SKL mechanism using CRT shown that the number of recalculations of keys are reduced when files are added to or deleted from the system. However for a particular user when new files are added to the system still it is required to recalculate the key with the already existing files.

This research proposes a method to encode the key number derived using CRT. The Tribes of Gaussian or Rational Farey Fractions are used to encode the key number. Advantages of this encoding are –

i) It is not required to decode the key to determine the access rights

ii) It is not required to derive the key number for already existing files. This method increases the security level of the system by encoding the key using arbitrary Farey Fraction.