CHAPTER 5

RESULTS AND ANALYSIS

5.1 Results and Analysis of ED Process

5.1.1 Introduction to cryptanalysis

Cryptanalysis is the flip-side of cryptography: it is the science of violating authentication schemes, and in general, breaking cryptographic in order to design a robust encryption algorithm. Cryptographic cryptanalysis is to find and correct any weaknesses. This is precisely encryption algorithms are ones that have been made available to have been exposed to public scrutiny for years. The various techniques in cryptanalysis attempting to compromise attacks, some attacks are general, whereas others apply only to certain of the better-known attacks.

Part of this work is published/presented/submitted


5.1.2 Basic cryptanalytic attack

Cryptanalytic attacks are generally classified into six categories that distinguish the kind of information the cryptanalyst has available to mount an attack. The categories of attack are listed here roughly in increasing order of the quality of information available to the cryptanalyst, or, equivalently, in decreasing order of the level of difficulty to the cryptanalyst. The objective of the cryptanalyst in all cases is to be able to decrypt new pieces of ciphertext without additional information. The ideal for a cryptanalyst is to extract the secret key.

i) A ciphertext-only attack is one in which the cryptanalyst obtains a sample of ciphertext, without the plaintext associated with it. This data is relatively easy to obtain in many scenarios, but a successful ciphertext-only attack is generally difficult, and requires a very large ciphertext sample.

ii) A known-plaintext attack is one in which the cryptanalyst obtains a sample of ciphertext and the corresponding plaintext as well.

iii) A chosen-plaintext attack is one in which the cryptanalyst is able to choose a quantity of plaintext and then obtain the corresponding encrypted ciphertext.

iv) An adaptive-chosen-plaintext attack is a special case of chosen-plaintext attack in which the cryptanalyst is able to choose plaintext samples dynamically, and alter his or her choices based on the results of previous encryptions.
v) A chosen-ciphertext attack is one in which cryptanalyst may choose a piece of ciphertext and attempt to obtain the corresponding decrypted plaintext. This type of attack is generally most applicable to public-key cryptosystems.

vi) An adaptive-chosen-ciphertext is the adaptive version of the above attack. A cryptanalyst can mount an attack of this type in a scenario in which he / she have free use of a piece of decryption hardware, but is unable to extract the decryption key from it.

Note that cryptanalytic attacks can be mounted not only against encryption algorithms, but also, analogously, against digital signature algorithms, MACing algorithms, and pseudo-random number generators.

5.1.3 Exhaustive key search

Exhaustive key search, or brute-force search, is the basic technique of trying every possible key in turn until the correct key is identified. To identify the correct key it may be necessary to possess a plaintext and its corresponding ciphertext, or if the plaintext has some recognizable characteristic, ciphertext alone might suffice. Exhaustive key search can be mounted on any cipher and sometimes a weakness in the key schedule the cipher can help improve the efficiency of an exhaustive key search attack.

Advances in technology and computing performance will always make exhaustive key search an increasingly practical attack against keys of a fixed length. When DES was designed, it was generally considered secure
against exhaustive key search without a vast financial investment in hardware [DIFF77]. Over the years, however, this line of attack will become increasingly attractive to a potential adversary.

The current rate of increase in computing power is such that an 80-bit key should offer an acceptable level of security for another 10 or 15 years. In the mid-20s, however, an 80-bit key will be as vulnerable to exhaustive search as a 64-bit key is today, assuming a halved cost of processing power every 18 months. Absent a major breakthrough in quantum computing, it is unlikely that 128-bit keys, such as those used in IDEA and the AES, will be broken by exhaustive search in the foreseeable future.

5.1.4 Frequency Analysis

Simple substitution codes, such as the Caesar cipher, are vulnerable to attack based on frequency analysis using the known letter frequencies of English [STAL04]. For example, it is known that the most common letters, in order from most common, are E, T, A, O, I, N and S. When attacking the cipher text of a substitution code, one first determines the most common letters. For the cipher text ZNKINKIQO YOTZNKSGOR K, N, O, and Z each occur three times, while the others occur only once or twice. It is natural to assume that one of these most frequent symbols represents the letter e (see figure 5.1 and 5.2). Trying K = e and then guessing all letters are shifted by six in a Caesar code, causes everything to fall in place, producing the message, “The check is in the mail”. More complex substitution codes are designed to increase the number of possible keys and render frequency analysis less potent.
Cryptogram

Advanced substitution codes substitute letters or symbols of the cipher text alphabet according to an arbitrary pattern. That is, a general substitution cipher may represent the letter a by K, b by X, c by F, and so forth, with the correspondence being unique in each direction. Symbols other than letters can be used for the cipher alphabet.

Figure 5.1: Relative Frequency of English alphabets in decreasing values of Relative Frequency

In any case, there are 26 different symbols, each corresponding to a plaintext letter. If ordinary letters are used for the cipher alphabet, the code system can be described by a permutation of the 26 letters of the alphabet, as shown in the example below. Plaintext A B C D E F G H I J K L MN OP QR S T U V WX Y Z Cipher text F J WS N O BKM U E I H Z X C Q RD
A cryptogram is vastly more complex than a Caesar cipher or even a pigpen cipher, for while there are 26 possible Caesar cipher keys, corresponding to the 26 possible shifts of the alphabet, there are 26! ≈ 4 × 10^{26} possible cryptogram keys corresponding to the 26! different permutations of the alphabet. This is an enormous number of possibilities.

One approach to breaking a substitution code is by trial and error or brute force, trying each possible key until a result makes sense. We can imagine a fast computer applied to the problem of solving a cryptogram by this trial-and-error procedure. The computer would cycle through the possible permutations of 26 letters, checking if the result were reasonable. Assuming that the computer could check 100 million permutations per second (which is optimistic since there would be considerable effort to determine if the result were reasonable), it would take about 2 \times 10^{26}/10^8 = 2 \times 10^{18} seconds to check one-half of the permutations (which on average is all that would need to be checked). There are 60 \times 60 \times 24 \times 365 = 31536000 seconds in a year. So it would take 2 \times 10^{18}/ (0.31536 \times 10^8) \approx 6 \times 10^9 = 6 \text{ billion years to complete the computation.}

Despite the complexity of a general substitution code, it preserves much of the character of plaintext language. Letter frequencies are preserved, being merely translated to the substitute letters or symbols. If e is coded as K, then K will likely appear more frequently than any other letter in the cipher, and this will suggest that K is the substitute for e. Word
structure is also preserved. For example, double letters in plaintext appear as double letters in the cipher text.

Figure 5.2: Relative Frequency of English alphabets in decreasing values of Relative Frequency

Example 5.1: (An important message).

To attack the cipher text

GFX XCXRUWKQJKWGJCDXDJCGFX FWVVGJGFX XBKG

First perform a frequency analysis, realizing that it may not be accurate for such a short message. The following counts are obtained
All the rest have counts of 1. Frequency analysis suggests $X = e$ and $G = t$. Then the fact that the word GFX appears three times suggests that it is *the*, the most common three-letter word. This gives $F = h$. We note that the two-letter word GJ starts with t under our assumption, and it is logical therefore to assume the word is *to*, which gives $J = o$. This means that the two-letter word JC starts with an o and hence it likely that $C = n$. The two-letter word WK contains no t, o, h, or n. Hence, a likely choice is is, which gives $W = i$ and $K = s$.

So the message is

```
G F X  X C X R U  W K  Q J K W G W J C X D
```

```
<table>
<thead>
<tr>
<th>t</th>
<th>h</th>
<th>e</th>
<th>n</th>
<th>e</th>
<th>i</th>
<th>s</th>
<th>o</th>
<th>s</th>
<th>i</th>
<th>t</th>
<th>i</th>
<th>o</th>
<th>n</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>F</td>
<td>X</td>
<td>X</td>
<td>C</td>
<td>X</td>
<td>R</td>
<td>U</td>
<td>W</td>
<td>K</td>
<td>Q</td>
<td>J</td>
<td>K</td>
<td>W</td>
<td>G</td>
</tr>
</tbody>
</table>
```

From here it is easy to fill in the missing letters to obtain the message: the enemy is positioned on the hill to the east. This approach has been duplicated in a computer program for solving cryptograms, which includes a dictionary of the 1,000 most common words in English, partitioned into words of different lengths and different structure. For example THAT, HIGH, and AREA are in the same group because the first and fourth letters agree in each of these words. The method systematically tries letter assignments in an attempt to maximize the number of words that match those in the dictionary. Cryptograms of about 30 letters in length
appear in puzzle books as challenges, and most can be easily solved by hand in half an hour or so.

**Justification on Frequency Analysis**

Among so many cryptanalytic techniques, frequency analysis or frequency count is the most basic one other than brutal force. Frequency analysis analytical frequency analysis technique by using the monogram first, then the two-letter word, three-letter word and so on.

Then the concept of two-letter word gave rise to the concept of contacts of letters. Nevertheless, the frequency analysis techniques using monogram and contacts of letters become the most universal, most basic cryptanalytic procedures. It is a prerequisite for understanding all subsequent substitution cryptanalysis techniques.

Subsequently, a new cryptographic field called as public key cryptosystem or modern cryptography was introduced by Diffie and Hellman in 1976 [DIFF76]. In practical life nowadays, both classical cryptography and modern cryptography are used together under a hybrid scheme, e.g. IBM mainframes, to gain an optimization between the key security and computational load. Public key cryptosystem is used for exchanging the master keys.

This is applicable and conveniently implemented with the handy help of nowadays computing facilities. The only difference is just that instead of the 26 elements of alphabetical characters, now we have only two elements, i.e. binary “1” and “0”. Although it looks more complicated, it is
basically the same. It is just like the analogy of numerical computations between binary, decimal and hexadecimal numerical systems. Hence the frequency analysis remains as the prevailing basic requisite for cryptanalysis techniques.

Drawbacks of conventional Frequency Analysis

At a first look into the sentences that describe the frequency analysis on monogram, the method shall work theoretically for sufficient long texts. However, it is found in practical tests that the frequency distribution of English language is a science fiction.

This is due to the fact that the English texts in different circumstances deviate between fields, e.g. military, diplomatic, commercial, legal, literary, etc. Besides, different person may use different words at different frequencies. In addition, an energetic language is alive and always grows with time. The vocabulary is getting more and more abundant. It may even undergo mutation due to the enrichment of foreign words.

Hence these factors may cause the frequency fluctuations for the alphabetic letters in the texts. The fluctuations will then create the crossover problems when we try to match the empirical frequency counts with the ideal frequency counts.
5.1.5 Cryptanalysis of ED Process

Cryptanalysis is an art of analyzing and breaking ciphers; it is an attempt to take cipher text and produce the plaintext or, better yet, the key. The term attack in this context has the following implication: In intuitive terms a (passive) attack on a cryptosystem is any method of starting with some information about plaintexts and their corresponding cipher texts under some (unknown) key, and figuring out more information about the plaintexts. This section describes the possible attacks on the proposed ED Process stream cipher and consequences.

In this research, the most common methods are applied in the cryptanalysis of a Stream cipher algorithm; the following attacks are performed for this proposed Stream Ciphering.

**Cipher text only attack:**

An attack against (i.e., an attempt to decrypt) cipher text when only the cipher text itself is available (i.e., there is neither known plaintext nor key associated with the cipher text). Given only some information about n cipher texts, the attack has to have some chance of producing some information about the plain texts. As there is no linear or any other kind of mathematical relationship between the n different cipher texts, Cipher text only attack is not effective. However the set of pseudorandom block keys are used for set of plain text for ciphering there is possibility of deriving the natural number by working on all possible permutations of pseudorandom block keys.
Known plaintext attack:

In this cryptanalysis the attacker knows or can guess the plaintext for some parts of the cipher text. The task is to decrypt the rest of the cipher text blocks using this information, for example frequency analysis. However frequency analysis of the cipher text (section 5.1.6 - discussed in detail) it is shown that frequency analysis reveals very feeble information to the hacker to crack the cipher text or key.

Chosen plaintext attack:

In this attack, the attacker is able to have any text encrypted with the unknown key or guessing a key. The task in the chosen plaintext attack is to determine the key used for encryption. This research found that our ED Process is highly resistive to the cipher text only, known plain text, chosen Plaintext attacks.

Differential Cryptanalysis

Differential cryptanalysis attacks block ciphers which is the general method of attacking cryptographic algorithms. However in our method repeated application of the ED process thwarts an attempt to carry Differential cryptanalysis on the resultant cipher text.

Related Key Cryptanalysis

It is similar to differential cryptanalysis, but it examines the differences between keys. In this attack a relationship is chosen between a pair of keys, but does not know the keys themselves. It relies on simple relationship between sub keys in adjacent rounds, encryption of plain texts under both the original (unknown) key $K$, and some derived keys $k_1,k_2$..... It
is needed to specify how the keys are to be changed; there may be flipping of bits in the key without knowing the key. ED Process stream cipher admits several related-key attacks which arise from the severe simplicity of its key schedule.

5.1.6 Frequency Analysis of ED Process

The Frequency analysis of ED Process is divided into two parts. In the first part the cipher text is analyzed for number of occurrence of each cipher character and compared with the standard frequency of English alphabets to find the matching of characters with the plain text. In the second part relative frequency of each character is compared with the standard relative frequency and then compared with the standard algorithms.

i) Frequency analysis of Cipher text

In this test number of occurrence of each character in cipher text is counted. And then each character is compared and replaced with the equivalent standard relative English alphabets. This results in a reasonable skeleton of the message. A more systematic approach are also followed like, certain words known to be in the text or repeating sequence of cipher letters and try to deduce their plain text equivalents. The process is repeated to digraphs (diagrams) and trigrams (trigrams).

The entire process is repeated with changing the number of characters per blocks of plain text. For the test to be more effective,
the plaintext and hence the cipher text length is chosen more than 4000 characters.

The following figures (Figure 5.3a and Figure 5.3b) give the initial test results of the ED process algorithm.

Figure 5.3 a: Frequency Analysis of Cipher text with variable plain text block
From the above graphs it is evident that the Frequency analysis of Cipher text does not reveal much to the hacker as the percentage of match of characters is very low and increase in the number of plain text stream characters per block also does not change the percentage of match drastically and hence their exists a nonlinearity.

ii) **Relative Frequency analysis of Cipher text**

Another way of revealing the effectiveness of the algorithm is as shown in the figure 5.4. The graph is developed in the following way: The number of occurrences of each character in the text is
counted and divided by the number of occurrences of the letter e (the most frequently used letter) to normalize the plot. As the result, e has a relative frequency of 1, t of about 0.76 and so on for the plain text. The points on the x-axis correspond to the letters in order of decreasing frequency.

Figure 5.4: Relative Frequency Analysis of Cipher text

Figure 5.4, therefore shows the extent to which the frequency distribution of characters, which makes it trivial to solve substitution ciphers, is marked by encryption. The ciphertext plot for the ED process stream cipher is much flatter than that of the other cipher algorithms, and hence the cryptanalysis using cipher text only is almost impossible.
5.1.7 Distribution of Invertible elements

The distributions of invertible elements play a vital role in the ED process stream ciphering, as the keys for ciphering plain text streams are the pseudorandom blocks generated from the group of invertible elements modulo natural number.

![Number of Invertible elements (Natural Numbers range 1234500-1234600)](image)

**Figure 5.5a: Distribution of Invertible Elements for range natural number from 1234500 to 1234600**

From figure 5.5a and 5.5b it is clear that for a given range of natural number the numbers of invertible elements for each natural number are randomly distributed. The keys for ciphering are the pseudorandom blocks that are generated from the invertible elements. To know or guess the keys
used for ED process stream ciphering it is necessary to work on the number of pseudorandom blocks, which in turn are distributed quite randomly in natural. Hence working on identifying the keys is almost an impossible task. This result strongly supports the implementation of Claude Shannon confusion effectively in the proposed stream ciphering.

![No of Invertible elements](image)

**Figure 5.5b: Distribution of Invertible Elements for range natural number from 1234500 to 1234600**

5.1.8 Exhaustive Key search for ED Process

As discussed in chapter 5, section 5.1.3, exhaustive key search is a brute-force search. It is the basic technique of trying every possible key in turn until the correct key is identified. The proposed stream cipher has
possible exhaustive key search attack as the keys are limited (keys are pseudorandom blocks of invertible elements).

Let \( n \) be the natural number used for the ED process stream cipher and let \( k \) be the number of invertible elements in the group deduced from \( n \).

For example for \( n = 12345677 \) there are \( k = 11919937 \) invertible elements, now the keys for stream ciphering are the blocks of invertible elements that are generated with each block having 26 elements corresponding to the 26 letters.

The number permutations of blocks of invertible elements can be created are
\[
{\begin{array}{c}
\text{nPr} \Rightarrow \quad \text{kP}_{26} = \frac{K!}{(K-26)!} \\
= 9.62 \times 10^{183}
\end{array}}
\]

\( 9.62 \times 10^{183} \) numbers of different ways the pseudorandom blocks and hence the keys for encrypting the plain text can be created. This is an enormous number of possibilities.

Exhaustive key search is to breaking a substitution code by trial and error or brute force, trying each possible key until a result makes sense. Assuming a fast computer applied to the problem of solving a cryptogram by this trial-and-error procedure. The computer would cycle through the possible permutations of each block 26 invertible elements, checking if the result were reasonable.
Assuming that the computer could check 10,000 million permutations per second (100 GHz Speed - which is optimistic since there would be considerable effort to determine if the result were reasonable), it would take about –

\[
\frac{(9.62 \times 10^{183})}{(2 \times 10^{11})} = 4.8 \times 10^{172} \text{ seconds}
\]

to check one-half of the permutations (which on average is all that would need to be checked).

There are \(60 \times 60 \times 24 \times 365 = 31536000\) seconds in a year.

So it would take

\[
\frac{(4.8 \times 10^{172})}{(0.31536 \times 10^8)}
\]

\[= 1.52 \times 10^{165} \text{ Years are required to complete the computation.}
\]

Hence the exhaustive key search technique doesn’t yield almost anything to the hacker.

5.1.9 Conclusion

The aim of the presented work was to introduce a new method replacing by one-time-pad cipher which is used in current stream ciphers that is XORing the key with the plain text to achieve the corresponding cipher. The method was described and the ciphers that are generated by this method have been analyzed and discussed (see section 3.3 in chapter 3 and
Section 4.2 in chapter 4). Finally, the structure of ED Process - stream cipher implementation was explained in detail along with the algorithm. The presented work in implemented in VB6, figure 5.6 is the screen shot of the presented work.

![Figure 5.6: Screen shot of ED Process- a stream cipher](image)

All possible attacks on the presented algorithm are discussed; it is shown that the algorithms very simple and easy to implement and can with stand any type of attack. The chapter 6, section 6.1 gives the advantages and disadvantages of the proposed work.
5.2 Results and Analysis of Encoding HACM

5.2.1 Introduction

The following chapter gives the results of the work encoding the hierarchical of and their analysis. The proposed work is implemented with programming language VB6. Figure 5.7 and Figure 5.8 are the screen shots.

Figure 5.7: Screen shot of Verification of Access Rights for the user using the proposed work.

Part of this work is presented in

National Workshop on Cryptology 2007 (NWC-2007), Amrita Vishwa Vidyapeetham University, Coimbatore, Tamilnadu, India; 6th Sep to 8th Sep 2007. (Sponsored by Cryptology Research Society of India (CRSI))


As the proposed work is used to authenticate and allow the user to access the system, hence the response time for the same should be very fast, hence the time complexity of the system has to be analyzed.

The access control system has 3 algorithms associated with it

i) Chinese Reminder Theorem Algorithm to find the solution for the simultaneous equations. The Computing complexity of this algorithm is discussed in the section 5.2.1.
ii) Computing Gaussian Farey Fractions Algorithm. The computing complexity of this algorithm is discussed in the section 5.2.2.

iii) Storing and retrieval of the encoded key. The computing complexity of this algorithm is discussed in the section 5.2.3.

5.2.2 Computing complexity of Chinese Remainder Theorem (CRT)

Solve the following system of equations modulus 105 using CRT:

\[
\begin{align*}
    x &= 2 \pmod{3} \\
    x &= 3 \pmod{5} \\
    x &= 4 \pmod{7}
\end{align*}
\]

An efficient, polynomial-time algorithm to find the solutions to the Chinese remainder theorem is based on Euclid’s GCD algorithm, which is based on the following theorem:

If \( a = bq + r \), where \( b > 0 \), then \( \text{GCD} (a, b) = \text{GCD} (b, r) \).

**Example 5.1:** The following sequence of division steps demonstrates how to find \( \text{GCD} (2205, 195) \) using Euclid’s GCD algorithm:

Initially, let \( a = 2205, b = 195 \), then divide \( a \) by \( b \) to obtain the quotient and the remainder:
Step 1: \[ 2205 = 195 \times 11 + 60 \]
\[ \text{GCD} (2205, 195) = \text{GCD} (195, 60) \]

In each of the subsequent steps, the dividend and the divisor are based on the divisor and remainder, respectively, of the previous step. Thus, the subsequent steps are as follows:

Step 2: \[ 195 = 60 \times 3 + 15 \]
\[ \text{GCD} (195, 60) = \text{GCD} (60, 15) \]

Step 3: \[ 60 = 15 \times 4 + 0 \]
\[ \text{GCD} (60, 15) = \text{GCD} (15, 0) = 15 \]

Combining all these results yields \( \text{GCD} (2205, 195) = 15 \), which is the last divisor. Also, note that the above division process will always terminate because the remainder of each step is strictly smaller than its divisor (that is how division works in arithmetic), which means smaller than the previous remainder.

Further, it can be shown that the number of division steps in computing \( \text{GCD} (a, b) \) is \( \leq \lfloor 2 \lg M \rfloor + 1 \), where \( M = \max (a, b) \). (For example, \( M = \max (2205, 195) = 2205 \) in the above example, and \( \lfloor 2 \lg M \rfloor + 1 = 22 + 1 = 23 \).) This expression says the time complexity of Euclid’s GCD algorithm is \( O(\lg \max(a, b)) \), which is a polynomial-time algorithm in terms of the size of the two input integers (i.e., \( \lg a + \lg b \)).
One more useful result out of the GCD process is that for any two positive integers \( a \) and \( b \), the above algorithm can be extended to find integers \( t \) and \( u \) such that \( at + bu = \text{GCD}(a, b) \). We will demonstrate this extended Euclid’s algorithm using the above values of \( a \) and \( b \).

**Example 5.2:** From Step 2, it can be written as

\[
\text{Step 4: } \quad \text{GCD}(2205, 195) = 15 \\
= 195 - 60 \times 3
\]

Using Step 1,

\[
60 = 2205 - 195 \times 11,
\]

Which can substitute into step 4, replacing 60:

\[
\text{GCD}(2205, 195) = 195 - (2205 - 195 \times 11) \times 3 \\
= 2205 \times (-3) + 195 \times (1 + 33) \\
= 2205 \times (-3) + 195 \times (34)
\]

In general, the process starts with the second-to-last step writing the GCD as a “linear combination” of the equation’s dividend and divisor. Then, use the previous equation to solve for its remainder, and substitute this into the current result, giving a linear combination of the dividend and the divisor of the new equation. Repeating this process will yield a linear combination of the first equation’s dividend and divisor, i.e., of the form \( at + bu \), that is equal to the GCD \((a, b)\).
Next step is solving the system of equations.

First, find the GCD (5, 3) and write it as a linear combination of 5 and 3, using extended Euclid’s algorithm.

\[
\begin{align*}
5 &= 3 \times 1 + 2 \\
3 &= 2 \times 1 + 1 \\
2 &= 1 \times 2 + 0
\end{align*}
\]

Thus, \( \text{GCD} (5, 3) = 1 = 3 - 2 \times 1 = 3 - (5 - 3 \times 1) \times 1 = 5 \times (-1) + 3 \times 2 \).

Therefore, a solution to the first two equations –

\[
\begin{align*}
x &= 2 \pmod{3} \quad \text{and} \\
x &= 3 \pmod{5} \quad \text{is} \\
x &= 2 \times 5 \times (-1) + 3 \times 3 \times 2 \\
&= 18 - 10 \\
&= 8 \pmod{3 \times 5}.
\end{align*}
\]

Now solving equations:

\[
\begin{align*}
x &= 8 \pmod{15} \quad \text{and} \\
x &= 4 \pmod{7}.
\end{align*}
\]

Applying extended Euclid’s algorithm yields:

\[
\begin{align*}
15 &= 7 \times 2 + 1 \\
7 &= 1 \times 7 + 0
\end{align*}
\]

Thus, \( \text{GCD} (15, 7) = 1 = 15 \times 1 + 7 \times (-2) \).
Therefore, a solution (mod 15*7) is

\[
x = 8 \times 7 \times (-2) + 4 \times 15 \\
= -112 + 60 \\
= -52 \\
= 53 \pmod{105}
\]

Note that this process of solving a system of 3 equations is a polynomial time algorithm for arbitrary moduli \(m, n, p\) that are pairwise co-prime, since the complexity is

\[
O(\lg \max(m, n) + \lg \max(m, n, p)) = O(\lg \max(m, n, p)).
\]

### 5.2.3 Time and Space complexities of computing Farey sequence

For any positive integer \(n\), the Farey sequence of order \(n\) is the set of all irreducible fractions \(\frac{p}{q}\), with \(0 < p < q \leq n\), arranged in increasing order. An alternative definition could include \(\frac{0}{1}\) and \(\frac{1}{1}\) as special fractions.

For example, the Farey sequence for \(n = 5\) are given by:

\[
\begin{array}{ccccccccccc}
\frac{1}{5} & \frac{1}{4} & \frac{1}{3} & \frac{2}{5} & \frac{1}{2} & \frac{3}{5} & \frac{2}{3} & \frac{3}{4} & \frac{4}{5} \\
\end{array}
\]
The Farey sequence is a well-known concept in number theory, whose exploration has lead to a number of interesting results. Following are the related algorithms and their time and space complexities.

i. Algorithm to sort all unreduced fractions \( \frac{p}{q} \) and removing duplicates. The running time is \( O(n^2 \log n) \), which is almost optimal, but the space is \( O(n^2) \). Assuming that only fractions are generated, not storing them; otherwise, quadratic space is clearly the best possible.

ii. The space in the above algorithm can be reduced to \( O(n) \), without changing the running time. This uses a priority queue to merge \( n \) sequences, where the \( i^{th} \) such sequence is

\[
\frac{1}{i}, \frac{2}{i}, \ldots \frac{i-1}{i}
\]

iii. Algorithm to obtain the sequence of order \( n+1 \) from the sequence of order \( n \). Consider all consecutive fractions \( \frac{a}{b} \) and \( \frac{c}{d} \) from the sequence of order \( n \), and insert the mediant fraction \( \frac{a+c}{b+d} \) between them, if the denominator is \( n+1 \). This surprising construction is based on the initial observation made by Farey in 1816. The resulting algorithm is the worst so far: the running time is \( O(n^3) \), and the space is \( O(n^2) \).

iv. Algorithm to combine several properties satisfied by Farey sequence, one can get a trivial iterative algorithm, which generates the next Farey fraction, based on the previous two.
If \( \frac{p}{q} \) and \( \frac{p'}{q'} \) are the last two fractions, the next one is given by \( \frac{p''}{q''} \)

where

\[
p^* = \left\lfloor \frac{q + n}{q'} \right\rfloor \ p' - p \quad \text{and} \quad q'' = \left\lfloor \frac{q + n}{q'} \right\rfloor \ q' - q
\]

This is an ideal algorithm: it uses \( O(n^2) \) time and \( O(1) \) space.

v. The Stern-Brocot tree is obtained by starting with \( \frac{0}{1} \) and \( \frac{1}{1} \), and repeatedly inserting the mediant between any two fractions that are consecutive in the in-order traversal of the tree. Farey fractions form a sub tree of the Stern-Brocot tree, often called the Farey tree. Farey fractions are generated in order, by recursively exploring the tree. The algorithm requires quadratic time, and \( O(n) \) memory (corresponding to the maximum depth of the Farey fractions of order \( n \)).

![Farey tree](image)

**Figure 5.9: Fare tree**
5.2.4 Computing Complexity of Storing and Retrieval of the encoded key

In this method hierarchical structure is used to dictate the user storage of keys in the database. It is a tree structure where each node contains the calculated key as shown in the figure 4.9. All users are formed into different groups or departments according to a group table set up by the database administrator. When a user logs on to the database, he is assigned a user node according to his unique password which has been verified by a user table. The user node contains the calculated encoded key and a pointer that points to users own local splay tree. The local splay tree maintains file nodes which contain the file name and the unique lock number. This local splay tree contains the files that are accessible by the user. A global splay tree is also introduced to keep track of all files and their respective owners. Since the local splay tree maintains user accessible files only, a superior user can not find files that belong to his/her inferior users.

The intervening system then retrieves the owner-pointer of the file from the global file directory and compares the relationship between the two nodes. Figure 4.9 illustrates the hierarchical key storage structure with global and local splay tree. A common files system can also be maintained, that shall contain the files accessed by all the users.

When the system verifies the access right, it needs to retrieve both the encoded key and lock. If it is assumed that old single key lock system uses a splay tree to maintain lock numbers. Retriev ing one lock needs
$O(\log_2 N)$ processing time, if $N$ is the number of files in the system. However with the Local splay tree, retrieving on lock needs $O(\log_2 n)$ processing time, where $n$ is the average number of files in the local tree.

5.2.5 Conclusion

In this chapter the proposed access control system is analyzed. The solution is based on Chinese Remainder Theorem (CRT) and has two categories: encoding the key determined using CRT, encoding it with Gaussian Farey Fractions. The technique used in the proposed method increases the security level of the system as the encoding of the key acts as a thing layer of security and is visible only while encoding only.

This mechanism consists of very simple operations, which make it very efficient. Finally, we have utilized a set of experiments to verify our system; the experimental results provide evidence that supports our research.