An Approach to ensure Information Security through 252-Bit Integrated Encryption System (IES)

Saurabh Dutta[1], Jyotsna Kumar Mandal[2]

1. **Dr. B. C. Roy Engineering College, Durgapur-713206, West Bengal, INDIA**
   E-mail: ***saurabhdutta06061973@gmail.com***
2. **Kalyani University, Kalyani-741235, West Bengal, INDIA**
   E-mail: ***jkm.cse@gmail.com***

Abstract

In this paper, a block-cipher, Integrated Encryption System (IES), to be implemented in bit-level is presented that requires a 252-bit secret key. In IES, there exist at most sixteen rounds to be implemented in cascaded manner. RPSP, TE, RPPO, RPMS, RSBM, RSBP are the six independent block-ciphering protocols, that are integrated to formulate IES. A round is constituted by implementing one of the protocols on the output produced in the preceding round. The process of encryption is an interactive activity, in which the encryption authority is given enough scope of flexibility regarding choice of protocols in any round. However no protocol can be used in more than three rounds. Formation of key is a run-time activity, which gets built with the interactive encryption process proceeds. Results of implementation of all six protocols in cascaded manner have been observed and analyzed. On achieving a satisfactory level of performance in this cascaded implementation, IES and its schematic and operational characteristics have been proposed.

Key Words: encryption, security, protocol, recursive, key, cipher, cryptography

1 Introduction

The proposed cipher, Integrated Encryption System (IES), is a 252-bit private-key based 16-round block-cipher that is capable of working in interactive mode on bit-streams of up to around 1 MB size. Rounds in IES are constituted through six

independent encryption tools, which are being termed here as "*Building Blocks*". Six building blocks are termed as follows:

1. RPSP (Recursive Positional Substitution based on Prime-Nonprime of cluster)
2. TE (Triangular Encryption technique)
3. RPPO (Recursive Paired Parity Operation)
4. RPMS (Recursive Positional Modulo-2 Substitution technique)
5. RSBP (Recursive Substitutions of Bits through Prime-nonprime detection of sub-stream)
6. RSBM (Recursive Substitutions of Bits through Modulo-2 detection of sub-stream)

On the basis of proposed schematic and operational characteristics, IES executes the task of encryption, and consequently a 252-bit key is generated at run-time that is to be kept secret. The invulnerability of the key against possible attacks because of the long key-space and the underlying complexity of the 16-round mathematical approach followed during encryption make IES a reasonably strong cipher and compatible enough with existing private-key based ciphers in the world of cryptography [1] [4] [5] [9] [10].

Six building blocks are presented in section 2. A simple cascaded approach of implementing all building blocks is presented in section 3. Section 4 is a proposal of IES that includes its principles, schematic characteristics, and operational, along with a proposal of 252-bit key. Section 5 concludes the entire approach. References are stated in section 6.

2 Building Blocks of IES

Correctness of all building blocks is mathematically well-proved and they have two things in common; all are block-ciphers and all are to be implemented in bit-level. Discussion on how these building blocks are handled in IES is made in detail in section 4. In this section, the process of converting one bit-stream into the corresponding encrypted bit-stream through all six building blocks has been presented respectively in sections 2.1, 2.2, 2.3, 2.4, 2.5 and 2.6.

2.1 The Building Block, RPSP

RPSP involves in generating a cycle. A generating function, $g(s, t)$, used to generate a target block (t) from a source block (s), is applied for a block in the first iteration. The rules to be followed while applying the generating function $g(s, t)$ are as follows:

1. A bit in the position i ($1 <= i <= n$-2) in the block s becomes the bit in the position (n-i) in the block t, if (n-i) is a non-prime integer.

2. A bit in the position i ($1 <= i <= (n$-2)) in the block s becomes the bit in the position j ($1 <= j <= (n$-i-1)) in the block t, where j is the precedent prime integer (if any) of (n-i), if (n-i) is a prime integer.

3. A bit in the position n in the block s remains in the same position in the block t

4. A bit in the position (n-2) in the block s is transferred in the block t to the position unoccupied by any bit after rules 1, 2 and 3 are applied.

The same generating function is applied to all the subsequent blocks. After a finite number of such iterations, the source block is regenerated.

If all blocks are not taken of the same size, the number of iterations required to regenerate the source block itself for one block may not be the same for another block. These varying numbers of iterations are synchronized by evaluating the least common multiple (LCM) of all these numbers. If for all the blocks, iterations are applied for number of times exactly equal to the value of the LCM, it can be ensured that for each of the blocks its respective source block will be generated.

The value of the LCM is the total number of iterations required to complete the tasks of encryption as well as decryption. Therefore any intermediate value between 1 and the value of the LCM may be considered as the number of iterations performed during the process of encryption. For the purpose of decryption the same process is to be applied for the remaining number of times [4].

2.2 The Building Block, TE

We consider a block $S = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5\ \ldots\ s^0_{n-2}\ s^0_{n-1}$ of size n bits, where s^0_i = 0 or 1 for $0 <= i <= (n\text{-}1)$.

Starting from MSB (s^0_0) and the next-to-MSB (s^0_1), bits are pair-wise XORed, so that the 1st intermediate sub-stream $S^1 = s^1_0\ s^1_1\ s^1_2\ s^1_3\ s^1_4\ s^1_5\ \ldots\ s^1_{n-2}$ is generated consisting of (n-1) bits, where $s^1_j = s^0_j \oplus s^0_{j+1}$ for $0 <= j <= n\text{-}2$, $\oplus$ stands for the exclusive OR operation. This 1st intermediate sub-stream S^1 is also then pair-wise XORed to generate $S^2 = s^2_0\ s^2_1\ s^2_2\ s^2_3\ s^2_4\ s^2_5\ \ldots\ s^2_{n-3}$, which is the 2nd intermediate sub-stream of length (n-2). This process continues (n-1) times to ultimately generate $S^{n-1} = s^{n-1}_0$, which is a single bit only. Thus the size of the 1st intermediate sub-stream is one bit less than the source sub-stream; the size of each of the intermediate sub-streams starting from the 2nd one is one bit less than that of the sub-stream wherefrom it was generated; and finally the size of the final sub-stream in the process is one bit less than the final intermediate sub-stream. Figure 2.2.1 shows the generation of an intermediate sub-stream $S^{j+1} = s^{j+1}_0\ s^{j+1}_1\ s^{j+1}_2\ s^{j+1}_3\ s^{j+1}_4\ s^{j+1}_5\ \ldots\ s^{j+1}_{n-(j+2)}$ from the previous intermediate sub-stream $S^j = s^j_0\ s^j_1\ s^j_2\ s^j_3\ s^j_4\ s^j_5\ \ldots\ s^j_{n-(j+1)}$. The formation of the triangular shape for the source sub-stream $S = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5\ \ldots\ s^0_{n-2}\ s^0_{n-1}$ is shown in figure 2.2.2.

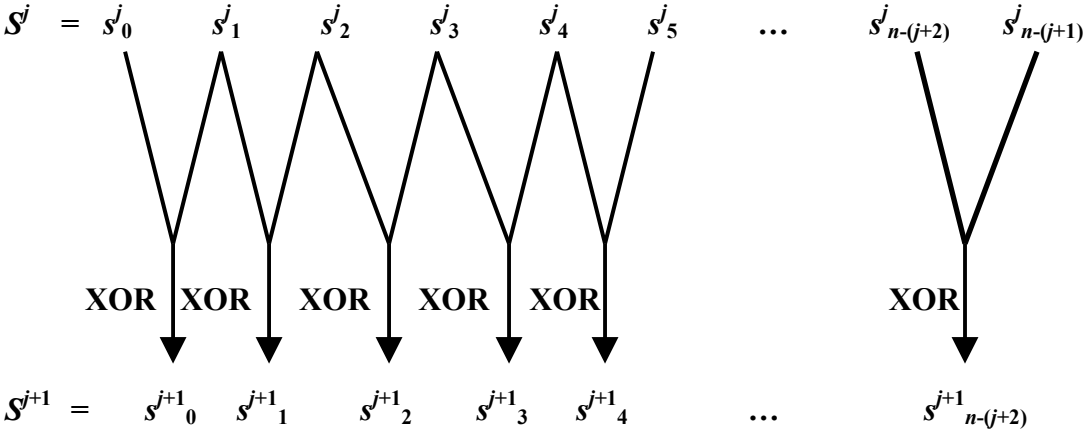


Figure 2.2.1
Generation of an Intermediate Sub-Stream in TE

$S = s^0_0 \quad s^0_1 \quad s^0_2 \quad s^0_3 \quad s^0_4 \quad s^0_5 \quad \ldots \quad s^0_{n-2} \quad s^0_{n-1}$

$S^1 = s^1_0 \quad s^1_1 \quad s^1_2 \quad s^1_3 \quad s^1_4 \quad \ldots \quad s^1_{n-2}$

$S^2 = s^2_0 \quad s^2_1 \quad s^2_2 \quad s^2_3 \quad \ldots \quad s^2_{n-3}$

$S^3 = s^3_0 \quad s^3_1 \quad s^3_2 \quad \ldots \quad s^3_{n-4}$

$S^4 = s^4_0 \quad s^4_1 \quad \ldots \quad s^4_{n-5}$

$S^5 = s^5_0 \quad \ldots \quad s^5_{n-6}$

$\ldots \ldots \ldots \ldots \ldots$

$S^{n-2} = s^{n-2}_0 \quad s^{n-2}_1$

$S^{n-1} = s^{n-1}_0$

Figure 2.2.2
Formation of a Triangle in TE

Corresponding to figure 2.2.2, a total of four options are available to form a target block from the source block $S = s^0_0 \; s^0_1 \; s^0_2 \; s^0_3 \; s^0_4 \; s^0_5 \; \ldots \; s^0_{n-2} \; s^0_{n-1}$. These options are shown in table 2.2.1. Figure 2.2.3 shows the same diagrammatically [3] [4] [6].

Table 2.2.1
Options for choosing Target Block from Triangle

Option Serial No.	Target Block	Method of Formation
001	$s^0_0 \, s^1_0 \, s^2_0 \, s^3_0 \, s^4_0 \, s^5_0 \ldots s^{n-2}_0 \, s^{n-1}_0$	Taking all the MSBs starting from the source block till the last block generated
010	$s^{n-1}_0 \, s^{n-2}_0 \, s^{n-3}_0 \, s^{n-4}_0 \, s^{n-5}_0 \ldots s^1_0 \, s^0_0$	Taking all the MSBs starting from the last block generated till the source block
011	$s^0_{n-1} \, s^1_{n-2} \, s^2_{n-3} \, s^3_{n-4} \, s^4_{n-5} \ldots s^{n-2}_1 \, s^{n-1}_0$	Taking all the LSBs starting from the source block till the last block generated
100	$s^{n-1}_0 \, s^{n-2}_1 \, s^{n-3}_2 \, s^{n-4}_3 \, s^{n-5}_4 \ldots s^1_{n-2} \, s^0_{n-1}$	Taking all the LSBs starting from the last block generated till the source block

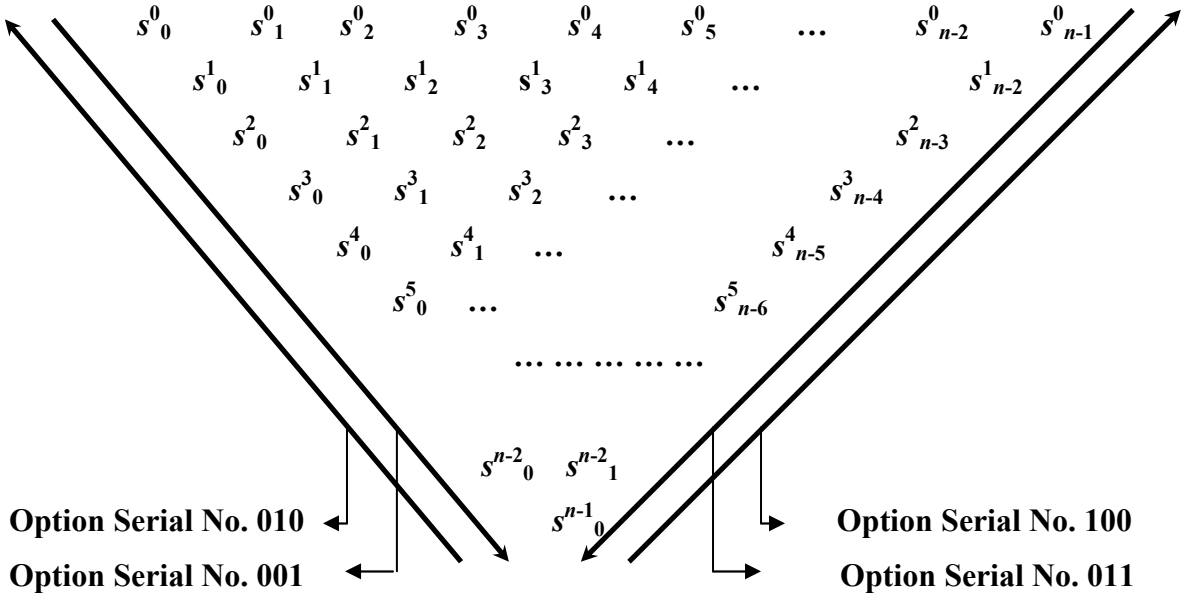


Figure 2.2.3
Diagrammatic Representation of Options for choosing Target Block from Triangle

2.3 The Building Block, RPPO

Let $P = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4 \ldots s^0_{n-1}$ is a block of size n in the plaintext. Then the first intermediate block $I_1 = s^1_0\ s^1_1\ s^1_2\ s^1_3\ s^1_4 \ldots s^1_{n-1}$ can be generated from P in the following way:

$s^1_0 = s^0_0$

$s^1_i = s^1_{i-1} \oplus s^0_i$, $1 <= i <= (n\text{-}1)$; $\oplus$ stands for the exclusive OR operation.

Now, in the same way, the second intermediate block $I_2 = s^2_0\ s^2_1\ s^2_2\ s^2_3\ s^2_4 \ldots s^2_{n-1}$ of the same size (n) can be generated by:

$s^2_0 = s^1_0$

$s^2_i = s^2_{i-1} \oplus s^1_i$, $1 <= i <= (n\text{-}1)$.

After this process continues for a finite number of iterations, which depends on the value of n, the source block P is regenerated.

If the number of iterations required to regenerate the source block is assumed to be I, the generation of any intermediate or the final block can be generalized as follows:

$s^{j}_{0} = s^{j-1}_{0}$

$s^{j}_{i} = s^{2}_{i-1} \oplus s^{j-1}_{i}$, $1 <= i <= (n\text{-}1)$; where $1 <= j <= i$.

In this generalized formulation system, the final block, which in turn is the source block, is generated when $j = i$.

Figure 2.3.1 pictorially represents this technique.

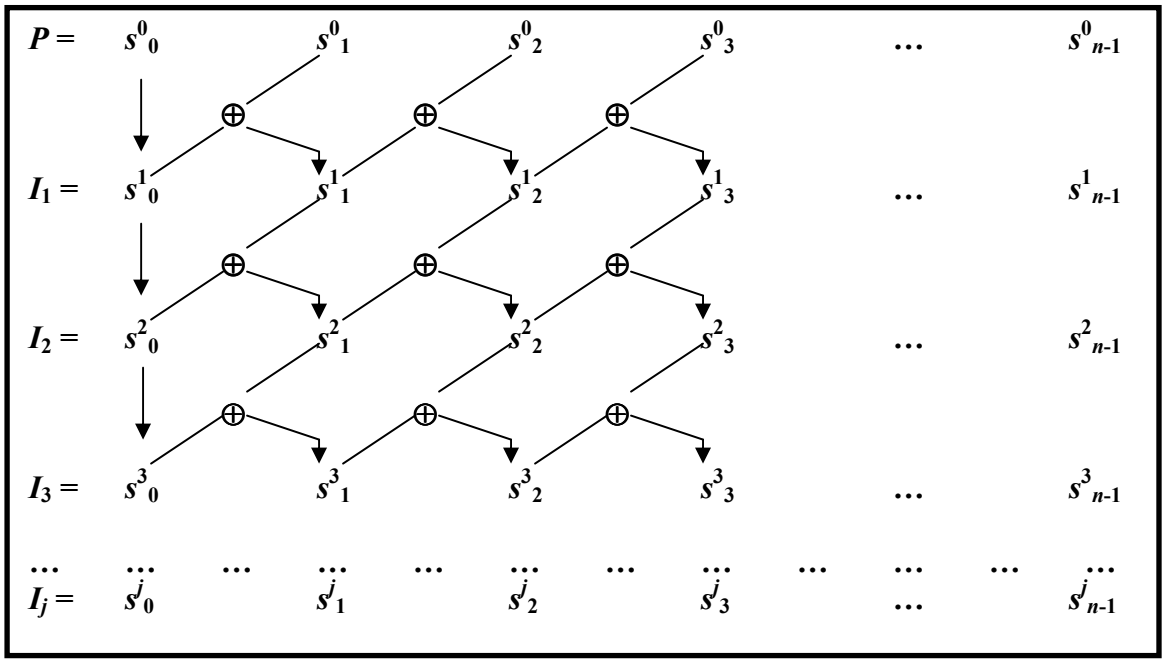


Figure 2.3.1
Pictorial Representation of the RPPO Technique

The entire scheme is a combination of encryption and decryption. Obviously, for the source block, *P*, if *I* iterations are required to complete the cycle, and the intermediate block generated after *N* iterations is considered as the encrypted block, (*I*-*N*) more iterations would be required to decrypt the encrypted block [2] [4].

2.4 The Building Block, RPMS

The algorithm to generate the encrypted block here can be presented through a pseudo code as follows [4] [6]:

Evaluate: D_L, the decimal equivalent, corresponding to the source block $S = s_0\ s_1\ s_2\ s_3\ s_4\ \ldots\ s_{L\text{-}1}$.

Set: $P = 0$.

LOOP:

Evaluate: Temp = Remainder of $D_{L\text{-}P} / 2$.

If Temp = 0

Evaluate: $D_{L\text{-}P\text{-}1} = D_{L\text{-}P} / 2$.

Set: $t_P = 0$.

Else

If Temp = 1

Evaluate: $D_{L\text{-}P\text{-}1} = (D_{L\text{-}P} + 1) / 2$.

Set: $t_P = 1$.

Set: $P = P + 1$.

If $(P > (L - 1))$

Exit.

ENDLOOP

2.5 The Building Block, RSBP

Following is the stepwise approach to be followed to encrypt the source bit-stream using this building block [1] [4] [9] [10]:

Step 1: Decompose the source stream, say, into a finite number of blocks, each preferably of the same size, say, L.

Step 2: Calculate the total number of primes and non-primes in the range of 0 to $(2^L\text{-}1)$. Accordingly, find minimum how many bits are required to represent each of these two numbers.

Step 3 to step 5 are to be applied for all the blocks.

Step 3: For the block under consideration, calculate the decimal number corresponding to that. Say, it is D.

Step 4: Find out if D is prime or nonprime. If D is a prime, the code value for that block is 1 and if not so, it is 0.

Step 5: In the series of primes or non-primes (whichever be applicable for D) in the range of 0 to (2^L-1), find the position of D. Represent this position in terms of binary values. This is the rank of this block.

After repeating these steps (3, 4 and 5) for all the blocks, following steps are to be followed.

Step 6: Say, there are N number of blocks. In the target stream of bits, put all the N code values one by one starting from the MSB position. So, in the target stream, the first N bits are code values for N blocks.

Step 7: For putting all the rank values in the target stream, we are to start from the N^{th} bit from the MSB position and then to come back bit-by-bit. Immediately after the N^{th} bit, put the rank value of the N^{th} block, followed by the rank value of the $(N-1)^{th}$ block, and so on. In this way, the rank value of the first block will be placed at the last.

Step 8: Combining all the code values as well as the rank values, if the total number of bits in the target stream is not a multiple of 8, then to make it so, at most 7 bits may have to be inserted. Insertion of these extra bits is to be started from the $(N+1)^{th}$ position. So, a maximum of 7 right shifting operations may have to be performed in the $(N+1)^{th}$ position, where that many 0's are inserted.

2.6 The Building Block, RSBM

Following is the set of steps to be followed for calculating the code value and the rank value of a certain block, and also to arrange these values to form the encrypted bit-stream [1] [4] [9] [10].

Step 1: For the block under consideration, say, of the length of L, calculate the corresponding decimal number. Say, it is D.

Step 2: Find out if D is odd or even. If D is odd, the code value for that block is 1 and if not so, it is 0.

Step 3: In the series of natural odd or even numbers (whichever be applicable for D) in the range of 0 to (2^L-1), find the position of D. Represent this

position in terms of binary values, which will require (L-1) bits. This is the rank of this block.

These three steps are to be repeated for all blocks, the number of which is, say, N. After finding all the N code values and the N rank values, those are to be integrated using the following set of steps.

Step 4: In the target stream of bits, put all the N code values one by one starting from the MSB position. So, in the target stream, the first N bits are code values for N blocks.

Step 5: For putting all the rank values in the target stream, we are to start from the N^{th} bit from the MSB position and then to come back bit-by-bit. Immediately after the N^{th} bit, put the rank value of the N^{th} block, followed by the rank value of the $(N\text{-}1)^{th}$ block, and so on. In this way, the rank value of the first block will be placed at the last.

3 A Simple Cascaded Approach of implementing all Building Blocks

This section provides a simple cascading approach of implementing all these six building blocks in a fixed sequence without having any special schematic or operational characteristics [4] [7] [8]. Figure 3.1 exhibits the steps followed during the encryption process. The sample file TLIB.EXE is encrypted through 6-phased cascading approach to generate FOX6.EXE.

Here the source file TLIB.EXE is first encrypted using the RPSP encryption technique with the unique size of each block is inputted as 8 bits. As the result, FOX1.EXE is generated.

FOX1.EXE is then encrypted using the TE encryption technique with the unique block size being fixed as 64 bits. As the result, FOX2.EXE is generated.

Next FOX2.EXE is encrypted for inputted unique block size of 16 bits using the RPMS encryption technique. As the result, FOX3.EXE is generated.

FOX3.EXE is then fed into the RSBP encryption technique with the fixed block size of 8 bits. As the result, FOX4.EXE is generated.

Then the RSBM encryption technique is used to encrypt FOX4.EXE using the fixed block size of 8 bits. As the result, FOX5.EXE is generated.

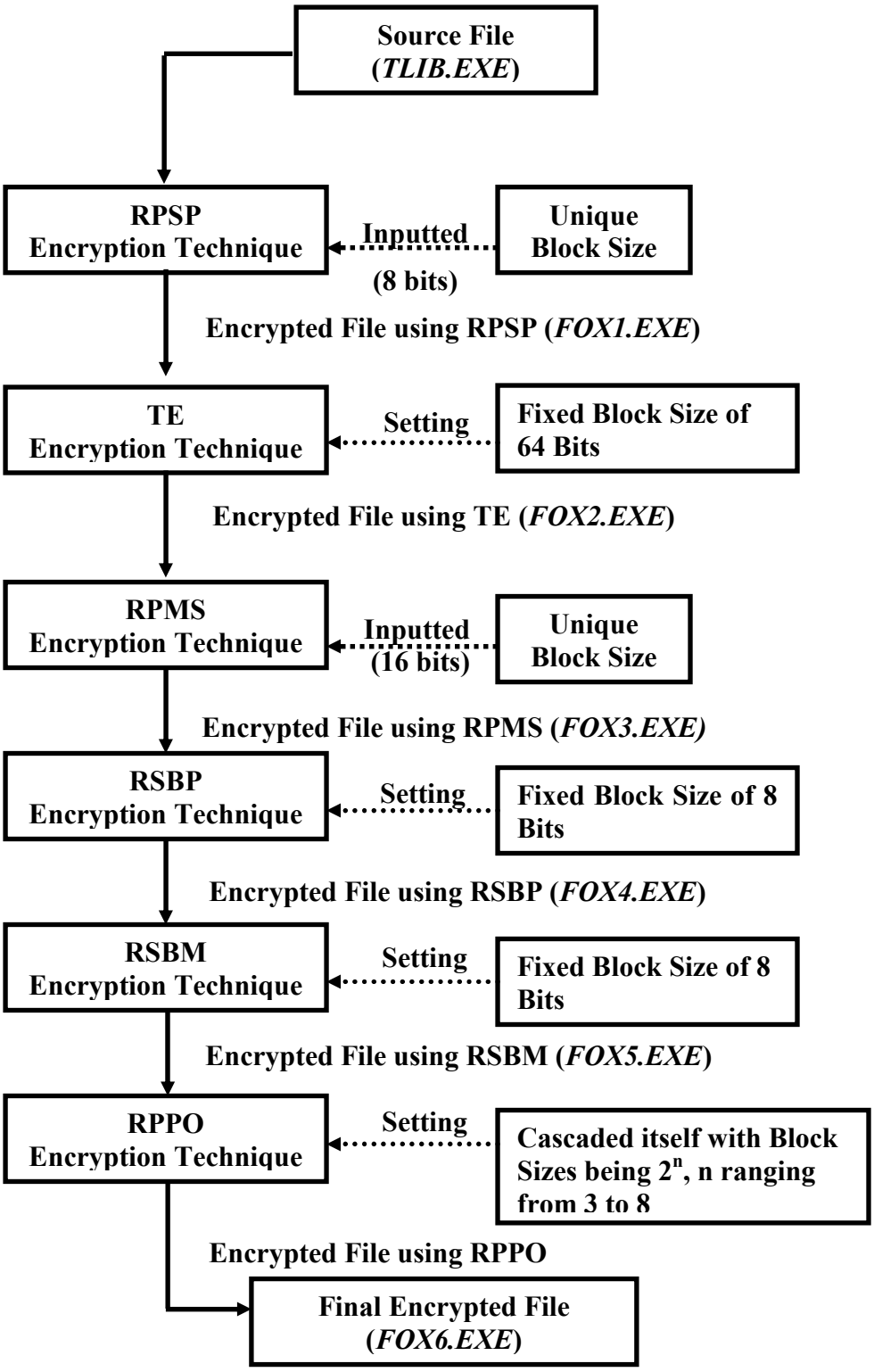


Figure 3.1
Steps followed during Cascaded Encryption

It has been analyzed and observed that even this simplest form of cascaded approach requires a 58-bit key for an error-free decryption.

In figure 3.2, in the structure of the 58-bit key, a total of 9 segments exist. Out of these, the first 6 segments, each with 3 bits, are used to identify the exact sequence of techniques followed during encryption. As there are 6 techniques, a 3-bit segment is required to identify any of those. The 7^{th} segment is of 20 bits of size, used to store the size of the source file in base-2 format, which is sufficient to accommodate the file size information of a file with 1.04 MB size approximately. This information is required while decrypting using the RSBP decryption technique. The final two segments, each of 10-bit size, are required to store the unique block sizes used during the implementation of the RPSP and the RPMS techniques. For the remaining four techniques, this information is not required here since those implementations have been done for fixed block sizes.

Table 3.1
Result of Cascaded Implementation

Source File (Size in Bytes)	File applying RPSP (C.S.V.)	File applying TE (C.S.V.)	File applying RPMS (C.S.V.)	File applying RSBP (C.S.V.)	Changed File (Size in Bytes)	File applying RSBM (C.S.V.)	File applying RPPO (C.S.V.)
UNZIP.EXE (23044)	F11.EXE (49310)	F12.EXE (18417)	F13.EXE (17338)	F14.EXE (26068)	24380	F15.EXE (22351)	F16.EXE (24489)
TCDEF.EXE (11611)	F21.EXE (63115)	F22.EXE (33517)	F23.EXE (37118)	F24.EXE (48593)	12171	F25.EXE (48206)	F26.EXE (53202)
WIN.COM (24791)	F11.COM (99157)	F12.COM (52643)	F13.COM (45311)	F14.COM (400444)	24283	F15.COM (222935)	F16.COM (171505)
KEYB.COM (19927)	F21.COM (78263)	F22.COM (59081)	F23.COM (49197)	F24.COM (121713)	20332	F25.COM (77737)	F26.COM (101565)
HIDCI.DLL (3216)	F11.DLL (12914)	F12.DLL (9583)	F13.DLL (8841)	F14.DLL (9221)	3433	F15.DLL (8446)	F16.DLL (9553)
PFPICK.DLL (58368)	F21.DLL (210525)	F22.DLL (174781)	F23.DLL (169556)	F24.DLL (189747)	61350	F25.DLL (192864)	F26.DLL (244145)
USBD.SYS (18912)	F11.SYS (135449)	F12.SYS (96371)	F13.SYS (95886)	F14.SYS (98424)	20140	F15.SYS (92551)	F16.SYS (118467)
IFSHLP.SYS (3708)	F21.SYS (15685)	F22.SYS (7162)	F23.SYS (6297)	F24.SYS (15258)	3889	F25.SYS (10257)	F26.SYS (10566)
ARITH.CPP (9558)	F11.CPP (10842)	F12.CPP (16595)	F13.CPP (12375)	F14.CPP (13045)	10056	F15.CPP (12906)	F16.CPP (12081)
START.CPP (14557)	F21.CPP (80174)	F22.CPP (53041)	F23.CPP (19420)	F24.CPP (40349)	15293	F25.CPP (47043)	F26.CPP (32577)

Apart from the 58-bit key-space, the results observed in this cascaded approach in terms of graphical layout of frequency distribution of characters in pairs of source and

encrypted files, Chi square values are inspiring. Table 3.1 is just an extract from all the results obtained in this regard. Here the term, C. S. V. stands for (Pearsonian) Chi Square Value.

All results shown in table 3.1 establish the fact that there exists no tendency of a steady progress in chi square value between the source file and the file last generated during encryption. In each case, the Chi Square value obtained is very high in comparison with the standard value. Hence it can be said that the intermediate files generated are heterogeneous in nature with 1% uncertainty, and as a result, it can be concluded that the cascaded scheme ensures high degree of security. Moreover, as it has been observed that even for this simple cascading, it requires a 58-bit key space to enable to have the correct decryption. Therefore it is easy to conclude that with the availability of having much more flexibility in the process of cascading, comparatively much longer key space may be generated. Accordingly, it was understood that offering more flexibility in the interactive encryption process will result in having a larger key-space, and hence to constitute a much more effective encryption system. To avail the following options, many more bits are to be added in the key space:

- Not to fix the unique block size; to input it only during the implementation for all techniques
- Allowing source size to be more than 1.04 MB
- Allowing one technique to be implemented more than once
- Allowing larger unique size for blocks
- Allowing varying block lengths at least for some techniques
- Allowing the choosing of option in implementing the TE encryption technique
- Allowing the choosing of the number of iterations to be performed during the process of encryption using the RPPO and the RPSP techniques

An integration of all six building blocks in more effective manner is what is attempted in IES.

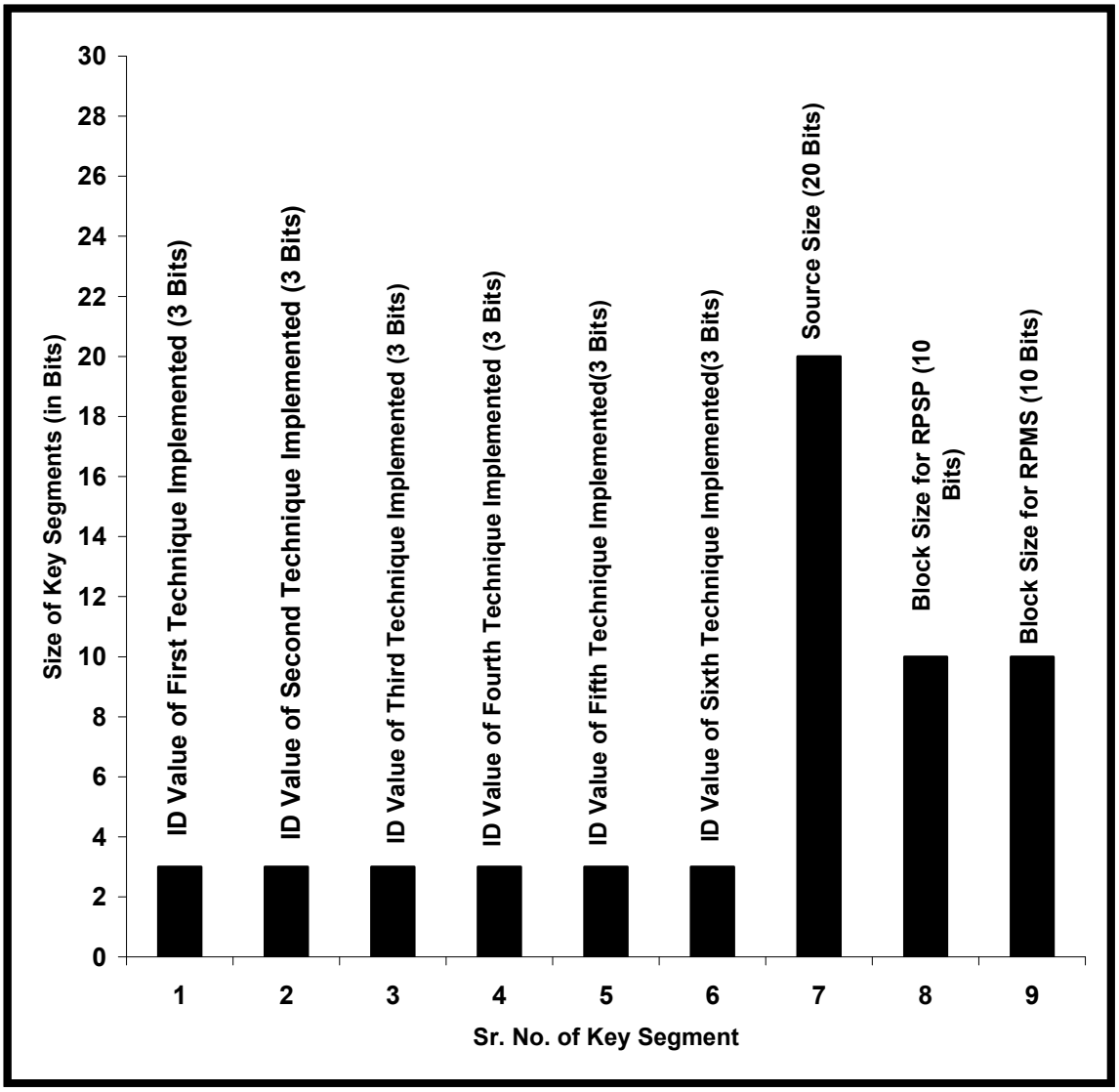


Figure 3.2
Structure of a 58-Bit Key with 9 Segments

4 The Proposal of Integrated Encryption System (IES)

This section presents a proposal of an integrated data encryption system on the basis of the principle of cascading using the six proposed techniques [4]. Section 4.1 points out the principles of the proposed system. Different schematic characteristics of the system are enlisted in section 4.2. Section 4.3 mentions different operational characteristics of the system. Section 4.4 presents the structure of the secret key to be used in the system.

4.1 Principles of IES

Following are the basic principles of the system:

- The fixation of the secret key is a run-time activity, during the process of encryption.
- Following the schematic restrictions (discussed in section 4.2) applicable to the encryption process, the task of encryption can be carried out and with each and every activity, the key space is upgraded
- During the process of decryption, the key generated at the end of encryption, is to be followed in each activity.

4.2 Schematic Characteristics of IES

The proposed system is a 16-level cascaded approach. Following points simplify the proposed concept to be followed during the process of encryption:

- Different proposed encryption techniques could be implemented altogether a maximum total of 16 times, although the encryption authority is given the flexibility to deserve the right to terminate the process of encryption after any number of implementations.
- Each proposed encryption technique could be implemented for a maximum 3 times, ranging from “no implementation at all” to 3 implementations.
- For each technique, during each implementation, a unique block length is to be considered and that should not exceed 512 bits. This length should depend on the choice of the encryption authority, so that it is to be inputted to the system.
- For each technique, during different implementations, different unique block sizes could be considered, but each of those must not exceed 512 bits.
- While using building blocks, RPSP or RPPO, the number of iterations to be performed during the process of encryption is exactly integral half of the total number of iterations required to complete the cycle.

4.3 Operational Characteristics of IES

During the operational phase of the encryption process, the system is to provide following facilities:

- At any point of time during the cascaded encryption, following steps are to be used sequentially to finalize a particular technique to be implemented:
  - The encryption authority selects a technique.
  - The system provides the corresponding chi square value between the original file and the file that would be generated if the technique is applied.
  - The encryption authority, may be on judging the value, either finalizes this implementation or cancels it to move to any other technique.
- With the finalization of each step during this cascaded implementation, the secret key, which is stored in a data file, is upgraded and the encryption authority is given the option to view the key at any point of time.
- If the encryption authority attempts to violet any of the schematic characteristics during the cascaded implementation, the system rejects this attempt by visualizing the reason of this rejection.
- The option for the termination of the cascading process is always to be available in the system, so that at any time the encryption authority can select this option, and, on confirmation, the termination can be granted.

4.4 Structure of the Secret Key for the Integrated System

Section 4.4.1 points out different criteria required to be fulfilled for the formation of efficient secret key. Section 4.4.2 is a discussion on the different proposed segments in the secret key. Finally, section 4.4.3 shows the proposed format for the secret key.

4.4.1 Criteria for an Efficient Key Generation

For the purpose of constructing a secret key, a perfect key management is needed, the objective of which is not to unnecessarily increase the key space, but to accommodate

all necessary information in the key in an efficient manner, so that the following set of criteria is satisfied:

- **No ambiguity:** There should be no mismatch between the schematic characteristics to be followed and the information stored in the key.
- **Easiness in accessing information:** All relevant information should be accommodated in the key as much adjacent as possible.
- **No repetition in providing information:** Directly or even indirectly there should not be repetition of any information.
- **Proper invalidation:** In certain situation, if any segment in the key becomes nonfunctioning, then that should be accommodated tactfully by perfectly invalid value.

Considering all these schematic characteristics and different criteria to be fulfilled, the structure of the 252-bit secret key has been proposed.

4.4.2 Formation of Different Segments in the 252-bit key

This section presents the way different segments in the 252-bit key are to be formed.

1. **Segment to identify the encryption technique:** Since there are 6 building blocks, a 3-bit segment is needed to identify a building block. For instance, table 4.4.2.1 shows one arbitrarily fixed set of ID values.

Table 4.4.2.1

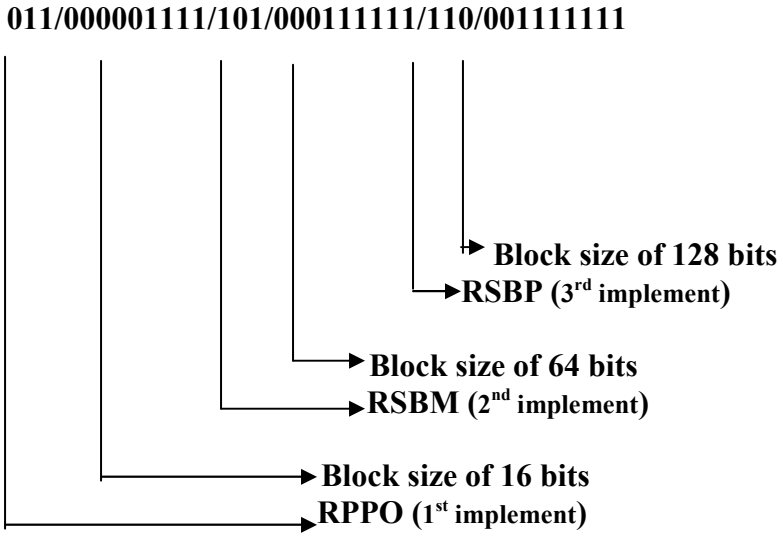
Arbitrarily chosen ID Values for Different Building Blocks

Building Block	Assigned ID Value	Building Block	Assigned ID Value
RPSP	001	RPMS	100
TE	010	RSBM	101
RPPO	011	RSBP	110

An ID value of “000” indicates implementation of no building block.

2. **Segment to identify Block Size:** As per the schematic characteristic, since the maximum block size can be 512 bits, a 9-bit segment is needed to express the unique block size used, with the assumption that the size of an *N*-bit block is represented by the 9-bit binary value corresponding to (*N*-1). For example, if the unique block size is 64 bits (*N* = 64), the 9-bit binary value expressing this size would be "001111111", which is the 9-bit binary value corresponding to 63 (*N*-1 = 63). The formation of this segment is based on the principle that the block size of 0 bit is impossible.

The 252-bit proposed key, starting from the MSB position, first accommodates the segment 1 (of 3 bits) and the segment 2 (of 9 bits) for the first encryption technique implemented, and since there can be maximum 16 implementations, altogether the first 16 x (3+9) = 192 positions are preserved for accommodating the values of these two segments for 16 implementations. "No implementation" is indicated by "000". For example, if there are only three implementations applied, in the sequence of RPPO, RSBM, and RSBP, with unique block sizes of 16 bits, 64 bits, and 128 bits respectively, then in the 192-bit structure, the first 3 x (3+9) = 36 bits using table 4.4.2.1, would be as the following:



Now, the first segment for each of the remaining probable 13 implementations is to be accommodated by "000" to indicate "no implementation" and hence, the respective

2nd segments are all immaterial, although it is advisable to put 9-bit null value ("000000000") in each of those segments, to avoid any sort of confusion.

3. **Segment to identify Source File Size:** During the process of decryption, the decryption authority must have the original file size to decrypt a file encrypted by applying the RSBP encryption technique. So, it is the responsibility during the process of encryption to fill in this 20-bit segment with the original file size.
4. **Segment to identify Size of File, on which RSBP is to be implemented second time:** Because each RSBP decryption technique requires the size of the file on which the corresponding RSBP encryption was implemented, another 20-bit segment is allocated for that purpose.
5. **Segment to identify Size of File, on which RSBP is to be implemented third time:** This 20-bit segment is allocated for the same reason as in 4. Since the RSBP encryption technique, like all other techniques, can be implemented at most 3 times, there is no need of any more segment for storing size of any intermediate file.

   Now, a 20-bit segment used for storing file size can store a file of size approximately up to 1.04 MB, and also the fact of "size alteration" following the RSBP encryption technique is to be considered. With this consideration, it can be pointed out that this structure of key would work successfully for a source file of up to 1 MB size.

4.4.3 Proposed Structure of 252-Bit Key for IES

On the basis of the discussion in section 4.4.2, figure 4.4.3.1 shows the proposed format.

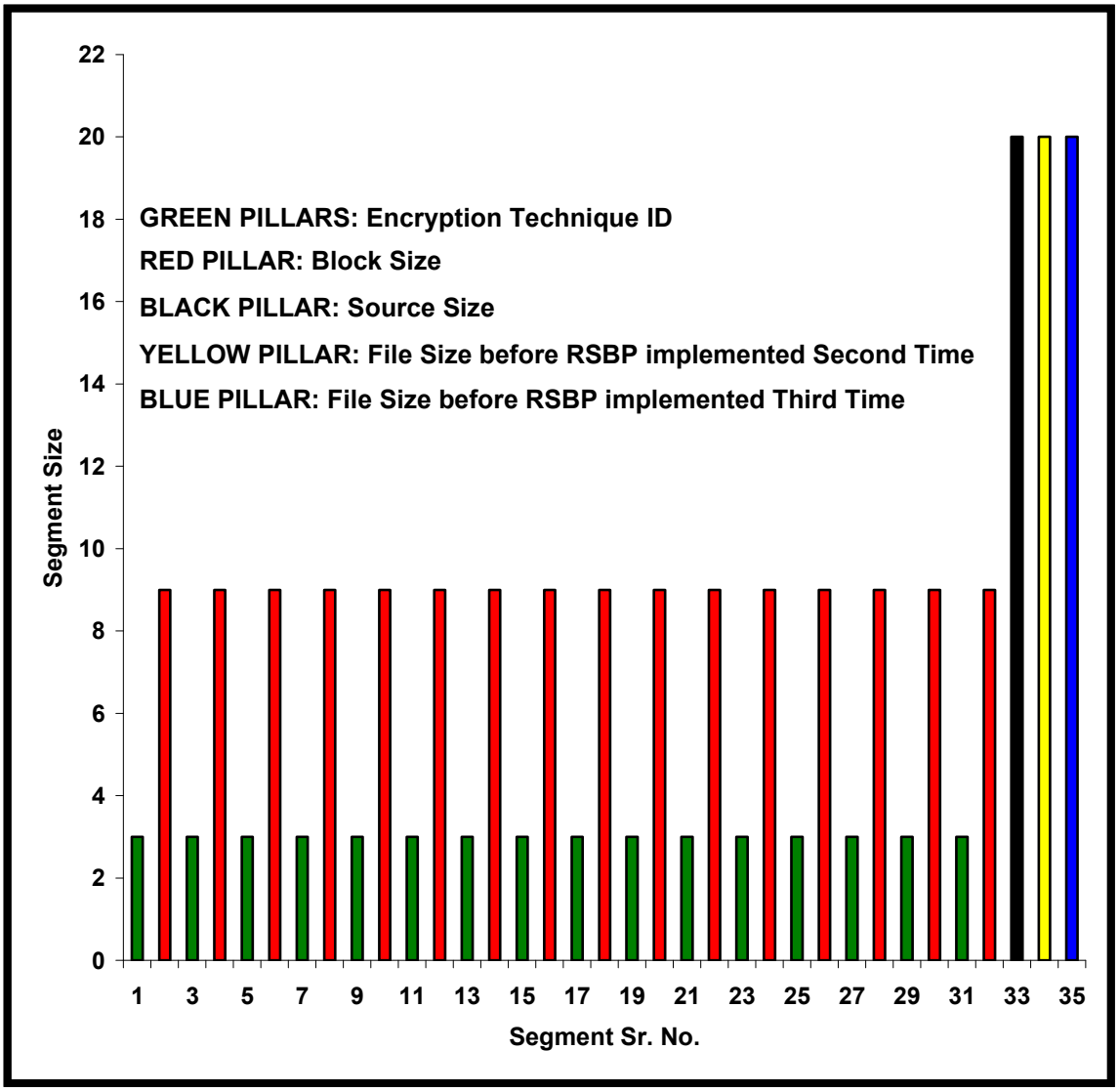


Figure 4.4.3.1
Proposed Format of 252-bit Secret Key with 35 Segments

5 Final Discussion and Conclusion on the Entire Approach

The approach of cascading plays a vital role in formulating an encryption system, as it involves a series of operations, which plays a leading role in enhancing security. The correct implementation of the proposed IES is expected to be an asset in the field of cryptography. Over times, in the field of cryptography, a number of ciphering protocols have been evolved. In terms of invulnerability against possible attacks to key, the proposed 252-bit IES is more secured than existing ciphers like 56-bit DES (Data Encryption Standard), 128-bit AES (Advanced Encryption Standard), 128-bit IDEA

(International Data Encryption Algorithm), or even 168-bit triple DES. The introduction of interactive mode while doing encryption offers more run-time involvement of the encryption authority. Consequently, in spite of using a ciphering protocol that is open to the world of cryptanalysis, the authority can apply a lot of its own jurisdiction in implementing the protocol.

6 References

1. Dauglas R. Stinson, Cryptography Theory and Practice, Third Edition, Chapman & Hall/CRC

2. Dutta S. and Mandal J. K., "A Space-Efficient Universal Encoder for Secured Transmission", Proceedings of International Conference on Modelling and Simulation (MS' 2000 – Egypt, Cairo, April 11-14, 2000, pp -193-201

3. Dutta S., Mal S., Mandal J. K., "A Multiplexing Triangular Encryption Technique – A move towards enhancing security in E-Commerce", Proceedings of IT Conference (organized by Computer Association of Nepal), 26 and 27 January, 2002, BICC, Kathmandu, pp 120-132

4. Dutta Saurabh, "An Approach Towards Development of Efficient Encryption Techniques", A Ph. D. Dissertation, The University of North Bengal, West Bengal, INDIA, 2004

5. Dutta Saurabh, Mandal Jyotsna Kumar, "Development and Analysis of a Ciphering Model through Recursive Positional Substitution based on a Prime-Nonprime of Cluster", Communicated to Association for the Advancement of Modelling and Simulation Techniques in Enterprises (AMSE, France), www. AMSE-Modeling.org

6. Mandal J K, Dutta S, "Ensuring E-Security using a Private Key Cryptographic System Following Recursive Positional Modulo-2 Substitutions", Asian Applied Computing Conference, October 29-31, Kathmandu, Nepal (Proceedings published as LNCS by Springer), pp – 324-332

7. Mandal J. K., Dutta S., "A 256-bit recursive pair parity encoder for encryption", Advances D -2004, Vol. 9 nº1, Association for the Advancement of Modelling and

Simulation Techniques in Enterprises (AMSE, France), www. AMSE-Modeling.org, pp. 1-14

8. Mandal J.K., Dutta Saurabh, Mal S., "Toward implementation of security and storage efficiency for geographical information systems", Advances B – 2004, Vol. 47 nº3, Association for the Advancement of Modelling and Simulation Techniques in Enterprises (AMSE, France), www. AMSE-Modeling.org, pp. 1-12

9. Schneier Bruce, Applied Cryptography, Protocols, Algorithms, and Source Code in C, John Wiley &Sons Inc., 1996

10. Stallings William, "Cryptography and Network Security", Third Edition, Pearson Education