CHAPTER - 2

Encryption Through Recursive Carry Addition (RCA)

Contents

2.1 Introduction

The proposed technique in this chapter named, Encryption Through Recursive Carry Addition or (RCA), is a secret-key cryptosystem [118].

The technique considers a message as binary string on which the Recursive Carry Addition or (RCA) is applied. A block of n bits is taken, where n varies from 8 to 256, as stream length from a continuous binary string. The basic characteristics of this technique is to use of a **carry (c)** generated by the binary addition method. For a particular length of block, the block itself is regenerated after a finite number of such iterations. Any of the intermediate blocks during this cycle is considered to be the encrypted block. To decrypt the encrypted block from the cipher text, the same process is to be followed.

To achieve the security of a satisfactory level, it is proposed that different blocks or blocks should be of different sizes. Accordingly, for different blocks, number of iterations during the encryption and the number of iterations during the decryption also not necessarily should be fixed. This information in a proposed fixed format constitutes the secret key for the system, which is to be transmitted by the sender to the receiver, either with the message or in an isolated manner.

The technique does not cause any storage overhead. It provides a large key space, so that the chance of breaking the cipher text is almost nullified by any technique of cryptanalysis. The implementation on practical scenario is well proven with positive outcome.

Section 2.2 of this chapter describes the scheme of this technique with simple examples. Section 2.3 shows a simple implementation of the technique, where a sample text message has been considered for the purpose of transmission using the encryption. Section 2.4 gives the results obtained after implementing the RCA technique on a number of real-time files of different categories like *.EXE*, *.DOC*, *.DLL*, *.SYS* and *.CPP* files and it also presents a comparative result with RSA and TDES. Section 2.5 is an analytical presentation of the technique, where the RCA technique has been analyzed from different perspectives. Section 2.6 draws a conclusion on the technique.

2.2 The Scheme

The total message can be considered as blocks of bits with different size like 4 /8 /16/ 32/ 64/ 128/ 256. The rules to be followed for generating a cycle are as follows:

1. Consider any source stream of a finite number (where $N=2^n$, n =2 to 8).
2. Add 1(one) to the left most bit of the string with binary addition method. If there is carry, add this carry to the second bit of the source string and continue up to the last bit to get the first intermediate block.
3. Again add 1(one) to the second bit of the source stream with the binary addition method, and get the second intermediate block with the same carry addition to its respective places, continued at the last.
4. This addition process is continued till the last bit of the given string. If there is a carry at the last bit then simply add that carry to the first place of the respective iteration and continued up to the last, which is shown in the figure 2.1 below.

The process of encryption is the sub set of the entire set of work to form the cycle for a given block. Any intermediate block during the process of forming the cycle may be considered as the encrypted block. Hence if the number of iterations required to form the cycle is I for a block of size n, the number of intermediate blocks generated in the cycle is (I-1), because each iteration generates one block and the final iteration generates the target block, which, in turn, is the source block. Therefore if the p^{th} block formed in the cycle is considered to be the encrypted block, p may vary from 1 to (I-1).

Like in encryption, the process of decryption is also a sub set of the entire set of work required to form the cycle for the block. The only difference is that one intermediate block of the cycle (the encrypted block) is considered to be the source block in the process of decryption. If the block generated after the p^{th} iteration in the cycle (1 <= p <= (I-1), I being the total number of iterations required to form the cycle) is considered to be the encrypted block, number of iterations required to decrypt the encrypted block is (I-p).

Consider the block S = 1101 of size 4 bits. The flow diagram to show how positions of the bits of S and the different intermediate blocks can be reoriented with the carry values to complete the cycle is shown in figure 2.1. Figure 2.2 depicts the different

intermediate blocks and the target block generated during the encoding process. In the figure **c** stands for **carry**.

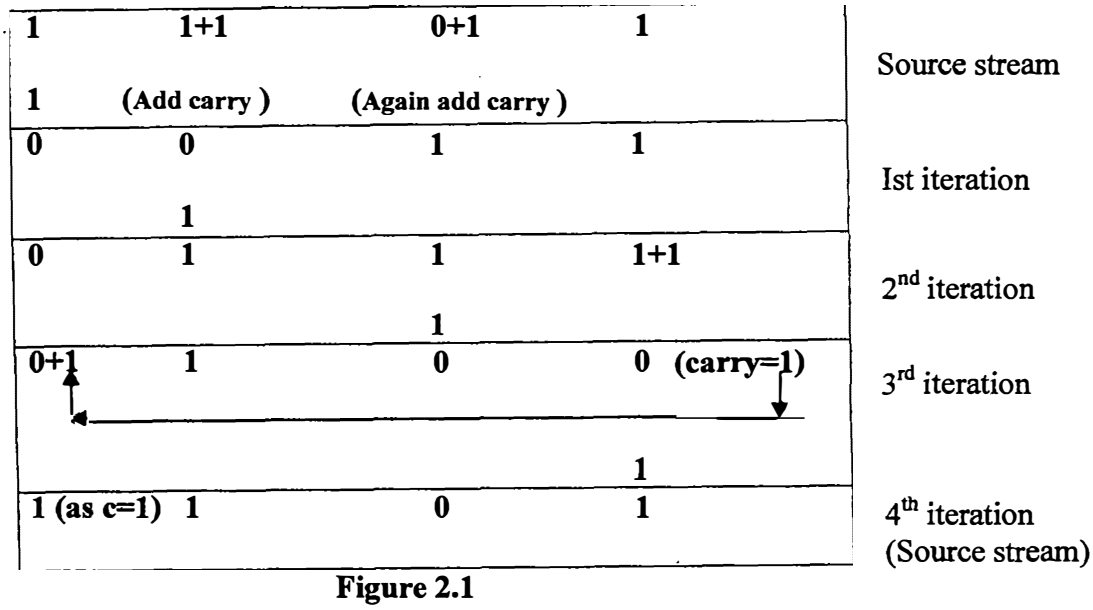

Figure 2.1
Forming the cycle for block [1101] for RCA technique

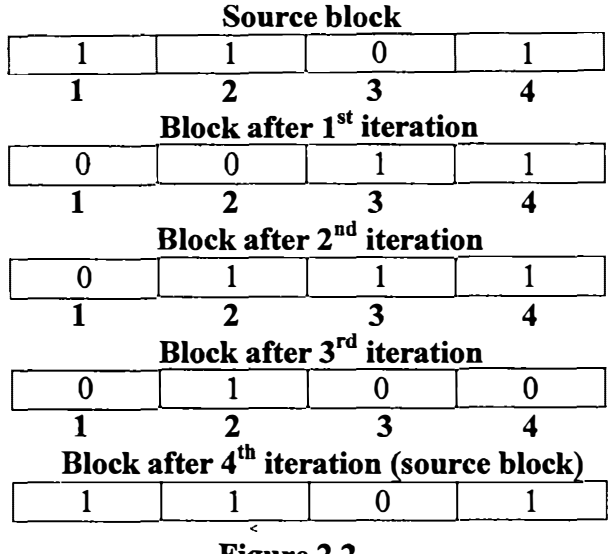

Figure 2.2
Different intermediate and target blocks generated in forming the cycle for block [1101]

2.3 Implementation

In this section, consider a simple text message that is to be transmitted by encrypting using RCA technique and after the transmission is over the encrypted message is to be decrypted using the same technique.

Consider the plaintext (P) as: Java Script. This stream is taken as the source stream.

Table 2.1 shows how each character in P can be converted into a byte.

Table 2.1
Character-to-byte conversion for the string "Java Script" for RCA technique

Character	Byte
J	01001010
A	01100001
V	01110110
A	01100001
<blank>	11111111
S	01010011
C	01100011
R	01110010
I	01101001
P	01110000
T	01110100

Combining all these bytes we obtained following source stream of bits (S) consisting of 88 bits, where "/" is used as the separator between two consecutive bytes:

01001010/01100001/01110110/01100001/11111111/01010011/01100011/01110010/
01101001/01110000/01110100.

For the simplicity, consider block size as 8, the number of source blocks are 11. Now, the source blocks generated from the source stream are:

S_1 = 01001010 S_2 = 01100001 S_3 = 01110110 S_4 = 01100001

S_5 = 11111111 S_6 = 01010011 S_7 = 01100011 S_8 = 01110010

S_9 = 01101001 S_{10} = 01110000 S_{11} = 01110100.

It is seen that for the block of size 8, the number of iterations required to regenerate the source itself is 8. Table 2.2 to table 2.12 show the formation of cycles for blocks S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, and S_{11} respectively. Now, for each of the

blocks, an arbitrary intermediate block, as indicated in each table, is considered as the encrypted stream.

Table 2.2
Formation of cycle for block S_1 for RCA technique

Source block	01001010
Block (I_{11}) after iteration 1	11001010
Block (I_{12}) after iteration 2	10101010
Block (I_{13}) after iteration 3	10011010
Block (I_{14}) after iteration 4	10000110
Block (I_{15}) after iteration 5	10001110
Block (I_{16}) after iteration 6	10001001
Block (I_{17}) after iteration 7	10001011
Block (I_{18}) after iteration 8	01001010
(Source block)	
Encrypted bloc	

Table 2.3
Formation of cycle for block S_2 for RCA technique

Source block	01100001
Block (I_{21}) after iteration 1	11100001
Block (I_{22}) after iteration 2	10010001
Block (I_{23}) after iteration 3	10110001
Block (I_{24}) after iteration 4	10101001
Block (I_{25}) after iteration 5	10100101
Block (I_{26}) after iteration 6	10100011
Block (I_{27}) after iteration 7	01100000
Block (I_{28}) after iteration 8	01100001
(Source block)	
Encrypted block	

Table 2.4
Formation of cycle for block S_3 for RCA technique

Source block	01110110
Block (I_{31}) after iteration 1	11110110
Block (I_{32}) after iteration 2	10001110
Block (I_{33}) after iteration 3	10101110
Block (I_{34}) after iteration 4	10111110
Block (I_{35}) after iteration 5	10110001
Block (I_{36}) after iteration 6	10110101
Block (I_{37}) after iteration 7	10110111
Block (I_{38}) after iteration 8	01110110
(Source block)	
Encrypted bloc	

Table 2.5
Formation of cycle for block S_4 for RCA technique

Source block	01100001
Block (I_{41}) after iteration 1	11100001
Block (I_{42}) after iteration 2	10010001
Block (I_{43}) after iteration 3	10110001
Block (I_{44}) after iteration 4	10101001
Block (I_{45}) after iteration 5	10100101
Block (I_{46}) after iteration 6	10100011
Block (I_{47}) after iteration 7	01100000
Block (I_{48}) after iteration 8	01100001
(Source block)	
Encrypted block	

Table 2.6
Formation of cycle for block S_5 for RCA technique

Source block	11111111
Block (I_{51}) after iteration 1	10000000
Block (I_{52}) after iteration 2	11000000
Block (I_{53}) after iteration 3	11100000
Block (I_{54}) after iteration 4	11110000
Block (I_{55}) after iteration 5	11111000
Block (I_{56}) after iteration 6	11111100
Block (I_{57}) after iteration 7	11111110
Block (I_{58}) after iteration 8 (Source block)	11111111

Encrypted block

Table 2.7
Formation of cycle for block S_6 for RCA technique

Source block	01010011
Block (I_{61}) after iteration 1	11010011
Block (I_{62}) after iteration 2	10110011
Block (I_{63}) after iteration 3	10001011
Block (I_{64}) after iteration 4	10011011
Block (I_{65}) after iteration 5	10010111
Block (I_{66}) after iteration 6	01010000
Block (I_{67}) after iteration 7	01010010
Block (I_{68}) after iteration 8 (Source block)	01010011

Encrypted block

Table 2.8
Formation of cycle for block S_7 for RCA technique

Source block	01100011
Block (I_{71}) after iteration 1	11100011
Block (I_{72}) after iteration 2	10010011
Block (I_{73}) after iteration 3	10110011
Block (I_{74}) after iteration 4	10101011
Block (I_{75}) after iteration 5	10100111
Block (I_{76}) after iteration 6	01100000
Block (I_{77}) after iteration 7	01100010
Block (I_{78}) after iteration 8 (Source block)	01100011

Encrypted block

Table 2.9
Formation of cycle for block S_8 for RCA technique

Source block	01110010
Block (I_{81}) after iteration 1	11110010
Block (I_{82}) after iteration 2	10001010
Block (I_{83}) after iteration 3	10101010
Block (I_{84}) after iteration 4	10111010
Block (I_{85}) after iteration 5	10110110
Block (I_{86}) after iteration 6	10110001
Block (I_{87}) after iteration 7	10110011
Block (I_{88}) after iteration 8 (Source block)	01110010

Encrypted block

Table 2.10
Formation of cycle for block S_9 for RCA technique

Source block	01101001
Block (I_{91}) after iteration 1	11101001
Block (I_{92}) after iteration 2	10011001
Block (I_{93}) after iteration 3	10111001
Block (I_{94}) after iteration 4	10100101
Block (I_{95}) after iteration 5	10101101
Block (I_{86}) after iteration 6	10101011
Block (I_{97}) after iteration 7	01101000
Block (I_{98}) after iteration 8 (Source block)	01101001

Encrypted block

Table 2.11
Formation of cycle for block S_{10} for RCA technique

Source block	01110000
Block (I_{101}) after iteration 1	11110000
Block (I_{102}) after iteration 2	10001000
Block (I_{103}) after iteration 3	10101000
Block (I_{104}) after iteration 4	10111000
Block (I_{105}) after iteration 5	10110100
Block (I_{106}) after iteration 6	10110010
Block (I_{107}) after iteration 7	10110001
Block (I_{108}) after iteration 8 (Source block)	01110000

Encrypted block

Table 2.12
Formation of cycle for block S_{11} for RCA technique

Source block	01110100
Block (I_{111}) after iteration 1	11110100
Block (I_{112}) after iteration 2	10001100
Block (I_{113}) after iteration 3	10101100
Block (I_{114}) after iteration 4	10111100
Block (I_{115}) after iteration 5	10110010
Block (I_{116}) after iteration 6	10110110
Block (I_{117}) after iteration 7	10110101
Block (I_{118}) after iteration 8 (Source block)	01110100

Encrypted block

As indicated in tables 2.3 to 2.12, intermediate blocks I_{15} (10001110), I_{24} (10101001), $I_{3:}$ (10101110), I_{45} (10100101), I_{56} (11111100), I_{66} (01010000), I_{73} (10110011), $I_{8:}$ (10001010), I_{95} (10101101), I_{107} (10110001) and I_{115} (10110010) are considered as the encrypted blocks, so that these together form the encrypted stream as follows:
10001110/10101001/10101110/10100101/11111100/01010000/10110011/10001010/101 01101/10110001/10110010, "/" being used as only the separator.

Converting the bytes into the corresponding characters, the following text is obtained as the encrypted text, which is to be transmitted:

C = "¿k¿ÞB²m¸ni

Now, the process for the decryption is the same as the encryption. After converting the ciphertext C into a stream of bits, it is to be decomposed into again 8-bit blocks (C_1, C_2... C_7). For each block, the same process as the encryption process is to be followed but for varying number of iterations. For example, while decrypting the encrypted block C_1, the number of iterations should be 6 – 1 = 5, iterations. The same logic is to be applied for the remaining blocks. After obtaining the source blocks in this

way, they are combined together in the same sequence and thus the source stream of bit is obtained, from which the source text is regenerated.

2.4 Results

For the purpose of the practical implementation, RCA technique has been implemented on total of 100 files of different categories namely *.EXE*, *.DOC*, *.DLL*, *.SYS* and *.CPP* files, each category being 20 in number.

Section 2.4.1 presents report of the encryption/decryption times and the Chi square values. Section 2.4.2 presents result of frequency distribution tests. A comparative result with the RSA and TDES system is given in section 2.4.3.

2.4.1 Result for encryption/decryption time and Chi square value

Section 2.4.1.1 shows the result on *.EXE* files, section 2.4.1.2 shows the result on *.DOC* files, section 2.4.1.3 shows the result on *.DLL* files, section 2.4.1.4 shows the result on *.SYS* files and section 2.4.1.5 shows the result on *.CPP* files. In all the cases, the sample blocks are taken of the same size.

Section 2.4.1.6 provides a report on the results for the average Chi square values of different categories of sample files.

2.4.1.1 Result for *.EXE* files

Table 2.13 gives the result of implementing the RCA technique on different executable files. The result includes the source file size, the time for encryption and the time for decryption. From the table, it is clear that the encryption and the decryption time increase with the increment in the size of the source file.

Twenty executable files are taken. The size of the files varies from 32033 bytes to 4325428 bytes. The encryption time varies from 0.054945 seconds to 6.978022 seconds, whereas the decryption time also varies from 0.054945 seconds to 7.252747 seconds. The values of the Chi square test results vary from 126567 to 8257955 with the degree of freedom 255. From these Chi square values a high degree of non-homogeneity of each encrypted file is seen in comparison to the corresponding source file.

Table 2.13
Result of *.EXE* files for RCA technique

Source file	Source size (in bytes)	Encryption time (in seconds)	Output File name	Output file size(in bytes)	Decryption time (in seconds)	Chi square value	Degree of freedom
TLIB.EXE	32033	0.054945	DEO1.EXE	32033	0.054945	300224	255
PCLINK.EXE	47443	0.054945	DEO2.EXE	47443	0.054945	126567	255
TEMC.CPP	57776	0.054945	DEO3.EXE	57776	0.054945	370144	255
BRCC32.EXE	76320	0.109890	DEO4.EXE	76320	0.109890	929417	255
WAVTOASF.EXE	83024	0.109890	DEO5.EXE	83024	0.109890	981575	255
EXTRACT.EXE	93242	0.109890	DEO6.EXE	93242	0.109890	511094	255
KRNL386.EXE	127040	0.219780	DEO7.EXE	127040	0.164835	615273	255
UPDATE.EXE	160256	0.219780	DEO8.EXE	160256	0.219780	482329	255
UTLIDX.EXE	254360	0.329675	DEO9.EXE	254360	0.384615	880021	255
LINK.EXE	460901	0.714286	DEO10.EXE	460901	0.769231	5908235	255
AUTOCONV.EXE	570640	0.824176	DEO11.EXE	570640	0.934066	7233528	255
TD.EXE	773980	1.153846	DEO12.EXE	773980	1.263736	5295981	255
TC.EXE	1263536	1.923077	DEO13.EXE	1263536	2.087912	4548804	255
WINZIP32.EXE	1425471	2.032967	DEO14.EXE	1425471	2.307629	3226002	255
BCW.EXE	1469984	2.307692	DEO15.EXE	1469984	2.472527	6501097	255
TT6.EXE	1656320	2.417582	DEO16.EXE	1656320	2.637363	3580608	255
WWF4.EXE	1835769	2.637363	DEO17.EXE	1835769	3.021978	2349555	255
MIDTOWN.EXE	1973736	3.186813	DEO18.EXE	1973736	3.296703	5621612	255
SWISH200.EXE	3133495	5.054945	DEO19.EXE	3133495	5.219780	8257955	255
POWERPNT.EXE	4325428	6.978022	DEO20.EXE	4325428	7.252747	6959065	255

Here since the operations while encrypting and decrypting are almost the same, the times needed for the encryption and the decryption are of little or no difference.

Figure 2.3 depicts the relationship between the source file size and the encryption time for *.EXE* files. As shown in the graph in figure 2.3, the encryption time increases almost linearly with the source file size.

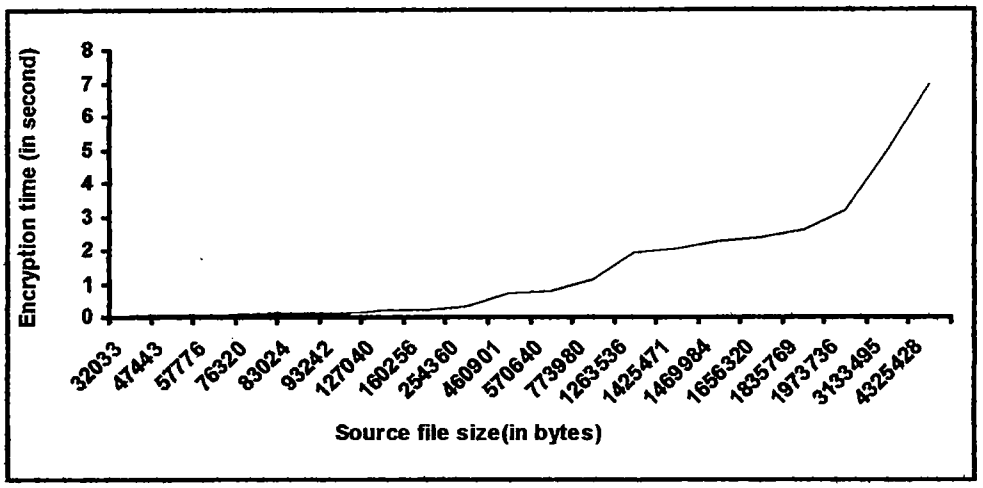

Figure 2.3
Graph to establish relationship between source file size and encryption time for *.EXE* files for RCA technique

2.4.1.2 Result for *.DOC* files

Table 2.14 shows the result of implementing the technique on 20 sample *.DOC* files. The size of the files varies from 106496 bytes to 4741120 bytes. The encryption time varies from 0.109890 seconds to 6.983542 seconds, whereas the decryption time also varies from 0.109890 seconds to 6.098901 seconds. The values of the Chi square test results vary from 61669 to 372610016 with the degree of freedom vary from 88 to 224. From these Chi square values a high degree of non-homogeneity of each encrypted file is seen in comparison to the corresponding source file.

Table 2.14
Result of *.DOC* files for RCA technique

Source file	Source size (in bytes)	Encryption time (in seconds)	Output file name	Output file size(in bytes)	Decryption time (in seconds)	Chi square value	Degree of freedom
QUESTION.DOC	106496	0.109890	DEO1.DOC	106496	0.109890	273599	224
TRIANGLE.DOC	129024	0.219780	DEO2.DOC	129024	0.164835	225896	224
LINUX.DOC	170520	0.219780	DEO3.DOC	170520	0.219780	632584	224
UTIL.DOC	199902	0.329670	DEO4.DOC	199902	0.274725	12380527	91
CLASSLIB.DOC	261879	0.384615	DEO5.DOC	261879	0.329670	30681776	107
UNIX6.DOC	387584	0.659341	DEO6.DOC	387584	0.494505	12619891	224
PROCESS1.DOC	527360	0.604396	DEO7.DOC	527360	0.659341	171720	224
UNIX1.DOC	700416	1.153846	DEO8.DOC	700416	0.934066	4144433	224
THIRTEEN.DOC	774144	1.214066	DEO9.DOC	774144	0.989011	77930	224
FINAL.DOC	909312	1.258681	DEO10.DOC	909312	1.208791	8535466	224
TWELVE.DOC	1037824	1.263736	DEO11.DOC	1037824	1.373626	61669	224
HISTORY.DOC	1282048	2.282747	DEO12.DOC	1282048	1.703297	444870	224
DEBUGREF.DOC	1655296	2.307692	DEO13.DOC	1655296	2.087912	7465953	224
ELEVEN.DOC	1829888	2.747253	DEO14.DOC	1829888	2.362637	94041	224
COURSE.DOC	1870448	2.747253	DEO15.DOC	1870448	2.417582	5960380	224
PART.DOC	2088961	3.571429	DEO16.DOC	2088961	2.802198	55526240	88
PEARL.DOC	4050014	6.538462	DEO17.DOC	4050014	5.000000	372610016	89
DOTNET.DOC	4110336	6.868132	DEO18.DOC	4110336	5.164835	3096224	224
MEMORY.DOC	4370944	6.983542	DEO19.DOC	4370944	5.659341	205144	224
FILEMMT.DOC	4741120	6.983542	DEO20.DOC	4741120	6.098901	969945	224

Figure 2.4 shows the graphical relationship between the source file size and the decryption time for *.DOC* files. As shown in the figure, the relationship is almost linear. This means that the decryption time increases linearly with the size of the source file considered for encryption.

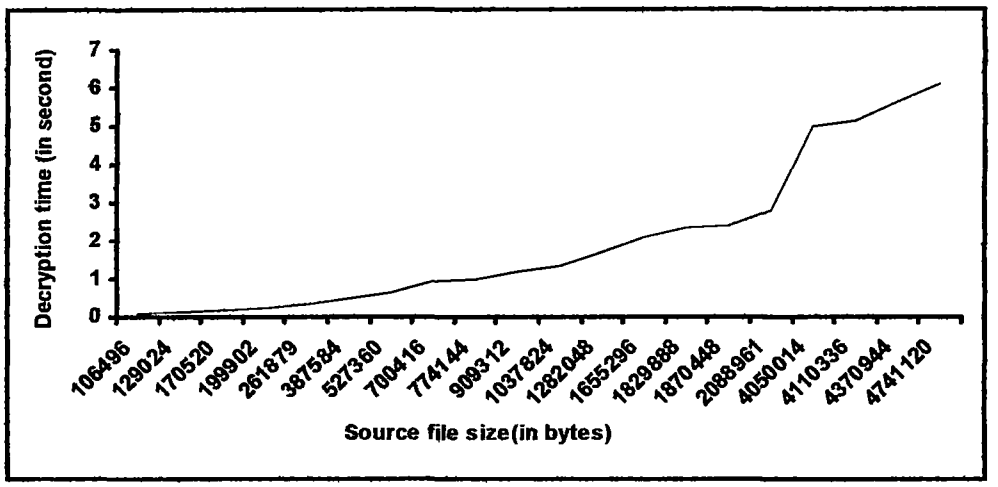

Figure 2.4
Graph to establish relationship between source file size and decryption time for *.DOC* files for RCA technique

2.4.1.3 Result for *.DLL* files

Table 2.15 gives the result of implementing the technique on 20 sample *.DLL* files. The size of the files varies from 28685 bytes to 5582897 bytes. The encryption time varies from 0.054945 seconds to 9.395604 seconds, whereas the decryption time also varies from 0.054945 seconds to 7.307692 seconds. The values of the Chi square test results vary from 347732 to 55551000 with the degree of freedom ranges from 206 to 254. From these Chi square values a high degree of non-homogeneity of each encrypted file is seen in comparison to the corresponding source file.

Table 2.15
Result of *.DLL* files for RCA technique

Source file	Source size (in bytes)	Encryption time (in seconds)	Output file name	Output file size(in bytes)	Decryption time (in seconds)	Chi square value	Degree of freedom
ENUMVAR.DLL	28685	0.054945	DEO1.DLL	28685	0.054945	347732	254
SELFREG.DLL	32256	0.054945	DEO2.DLL	32256	0.054945	102783	219
VCARD.DLL	36688	0.054945	DEO3.DLL	36688	0.054945	468712	254
ACTPANEL.DLL	51712	0.054945	DEO4.DLL	51712	0.054945	266242	254
WZCAB.DLL	65536	0.109890	DEO5.DLL	65536	0.109890	764611	254
ODBCINT.DLL	70982	0.109890	DEO6.DLL	70982	0.109890	4210088	206
VSAMI.DLL	81920	0.109890	DEO7.DLL	81920	0.109890	827770	254
MFC42.DLL	995383	1.648352	DEO8.DLL	995383	1.318681	8989041	254
OWL52.DLL	1107472	1.813187	DEO9.DLL	1107472	1.538462	8273230	254
BFC40.DLL	1187840	1.978022	DEO10.DLL	1187840	1.593407	9272143	254
DEVSHL.DLL	1347632	2.211758	DEO11.DLL	1347632	1.758242	8447326	254
COOLTYPE.DLL	1441792	2.417582	DEO12.DLL	1441792	1.923077	7228557	254
VGX.DLL	1753160	2.967033	DEO13.DLL	1753160	2.362637	3525923	254
ENGINE.DLL	2146304	3.406593	DEO14.DLL	2146304	2.802198	7060996	254
JVM.DLL	2461690	4.230769	DEO15.DLL	2461690	3.296703	9754395	254
MSENV.DLL	3383296	3.351648	DEO16.DLL	3383296	3.582418	6221269	254
MSO97RT.DLL	3673360	6.153846	DEO17.DLL	3673360	4.890110	2595513	254
JVM_G.DLL	4579395	7.857143	DEO18.DLL	4579395	6.043956	20628350	254
OUTLLIB.DLL	5275698	8.956044	DEO19.DLL	5275698	6.923077	11426359	254
MSO9.DLL	5582897	9.395604	DEO20.DLL	5582897	7.307692	55551000	254

Figure 2.5 shows the relationship between the source file size and the encryption/decryption time for these *.DLL* files. There exists almost linear relationship between these two parameters. This proves that for *.DLL* files also the encryption/decryption time increase linearly with the source file size.

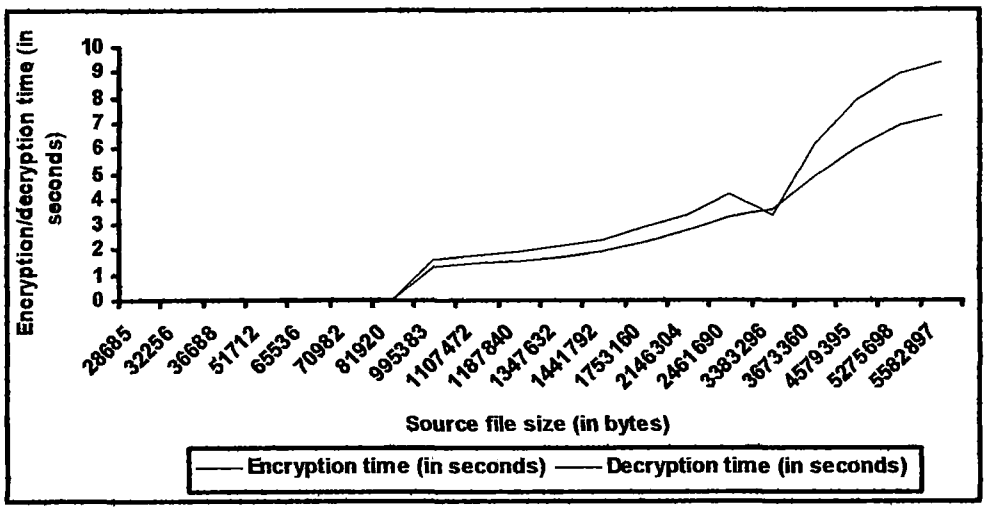

Figure 2.5
Graph to establish relationship between source file size and encryption/decryption time for *.DLL* files for RCA technique

2.4.1.4 Result for *.SYS* files

Table 2.16 shows the result after implementing the technique on 20 sample *SYS* files. The size of the files varies from 35392 bytes to 1748399 bytes. The encryption time varies from 0.054945 seconds to 3.021978 seconds, whereas the decryption time also varies from 0.054945 seconds to 2.307692 seconds. The values of the Chi square test results vary from 114958 to 8754148 with the degree of freedom 254. From these Chi square values a high degree of non-homogeneity of each encrypted file is seen in comparison to the corresponding source file.

Table 2.16
Result of *.SYS* files for RCA technique

Source file	Source size (in bytes)	Encryption time (in seconds)	Output file name	Output file size(in bytes)	Decryption time (in seconds)	Chi square value	Degree of freedom
NTIO412.SYS	35392	0.054945	DEO1.SYS	35392	0.054945	114958	254
I8O42PRT.SYS	45488	0.054945	DEO2.SYS	45488	0.054945	309313	254
DLC.SYS	56016	0.109890	DEO3.SYS	56016	0.054945	304224	254
NAVENG.SYS	65632	0.109890	DEO4.SYS	65632	0.109890	680246	254
LVCODEK.SYS	79120	0.109890	DEO5.SYS	79120	0.109890	239757	254
NBF.SYS	102160	0.164835	DEO6.SYS	102160	0.109890	938112	254
DMIO.SYS	135984	0.219780	DEO7.SYS	135984	0.164835	1137828	254
NETBT.SYS	148976	0.219780	DEO8.SYS	148976	0.164835	1773381	254
NWRDR.SYS	158608	0.274725	DEO9.SYS	158608	0.219780	1391272	254
NDIS.SYS	167760	0.274725	DEO10.SYS	167760	0.219780	1508606	254
SPCMDCON.SYS	187024	0.274725	DEO11.SYS	187024	0.219780	644123	254
SRV.SYS	242544	0.384615	DEO12.SYS	242544	0.329670	2272823	254
TCPIP.SYS	305520	0.494505	DEO13.SYS	305520	0.384615	2823472	254
DMBOOT.SYS	368240	0.549451	DEO14.SYS	368240	0.494505	2620915	254
NAVEX15.SYS	481216	0.824176	DEO15.SYS	481216	0.659341	4136534	254
NTFS.SYS	535248	0.934066	DEO16.SYS	535248	0.714286	7963321	254
VMODEM.SYS	704121	1.043956	DEO17.SYS	704121	0.934066	1078467	254
VPCTCOM.SYS	799826	1.153846	DEO18.SYS	799826	1.043956	2816246	254
WIN32K.SYS	1726256	2.967033	DEO19.SYS	1726256	2.307692	8754148	254
DIRECT.SYS	1748399	3.021978	DEO20.SYS	1748399	2.307692	8694071	254

Figure 2.6 establishes the graphical relationship between the source file size and the encryption time for *.SYS* files. It indicates that here also exists almost linear relationship between these two parameters. Therefore the encryption time increases almost linearly with the source file size.

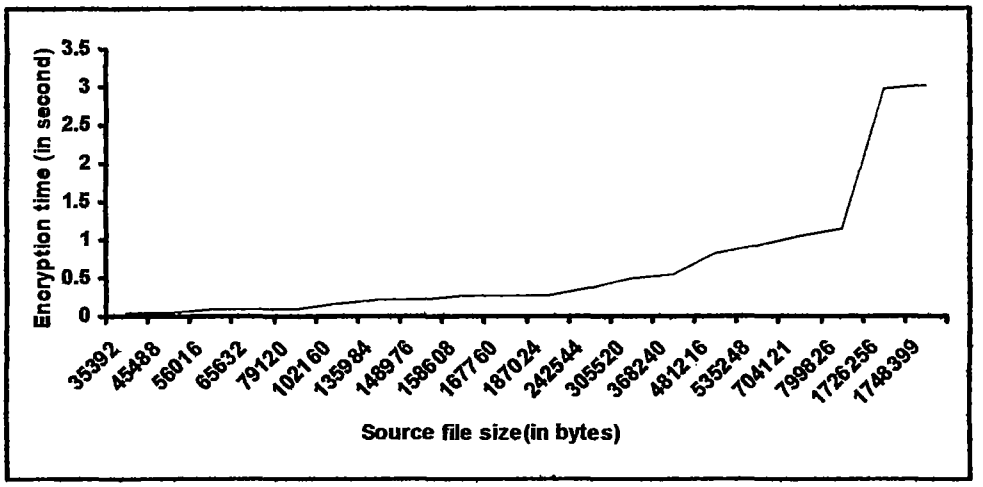

Figure 2.6
Graph to establish relationship between source file size and encryption time for *.SYS* files for RCA technique

2.4.1.5 Result for *.CPP* files

Table 2.17 summarizes the report of running the technique on 20 sample *.CPP* files. The size of the files varies from 30848 bytes to 404637 bytes. The encryption time varies from 0.054945 seconds to 0.656341 seconds, whereas the decryption time also varies from 0.054945 seconds to 0.494505 seconds. The values of the Chi square test results vary from 39447 to 194565600 with the degree of freedom vary from 84 to 91. From these Chi square values a high degree of non-homogeneity of each encrypted file is seen in comparison to the corresponding source file.

Table 2.17
Result of *.CPP* files for RCA technique

Source file	Source size (in bytes)	Encryption time (in seconds)	Output file name	Output file size(in bytes)	Decryption time (in seconds)	Chi square value	Degree of freedom
VIEWPREV.CPP	30848	0.054945	DEO1.CPP	29758	0.054945	568921	85
OLECLI2.CPP	41023	0.109890	DEO2.CPP	39447	0.054945	39447	84
OLECLI1.CPP	61600	0.109890	DEO3.CPP	59323	0.054945	59323	87
INET.CPP	72980	0.109890	DEO4.CPP	70101	0.109890	1173146	86
OCCSITE.CPP	89786	0.109890	DEO5.CPP	86262	0.109890	1561664	89
DBRFX.CPP	91269	0.164835	DEO6.CPP	87810	0.109890	2044262	85
WINCORE.CPP	109141	0.219780	DEO7.CPP	105194	0.109890	2659827	91
DBCORE.CPP	115208	0.219780	DEO8.CPP	110526	0.109890	4291539	89
DAOCORE.CPP	135431	0.219780	DEO9.CPP	129530	0.164843	5135549	89
BOOK.CPP	143336	0.274725	DEO10.CPP	142326	0.164843	694662	89
FLIGHT.CPP	154982	0.274725	DEO11.CPP	149114	0.164843	2633964	89
DRAMA.CPP	160850	0.274725	DEO12.CPP	149114	0.164843	2633964	90
GENIL.CPP	196836	0.329670	DEO13.CPP	189394	0.221978	4400968	90
DUIT.CPP	249132	0.439560	DEO14.CPP	246562	0.329670	1491142	89
COMMANDS.CPP	254630	0.439560	DEO15.CPP	254132	0.329670	7081902	80
PARSER.CPP	275583	0.439560	DEO16.CPP	265761	0.329670	6854556	90
DEAR.CPP	300565	0.549451	DEO17.CPP	300567	0.384615	194565600	86
COMP.CPP	377784	0.604396	DEO18.CPP	364915	0.494505	14971028	87
DOIT.CPP	390653	0.604396	DEO19.CPP	390653	0.494505	14971028	87
MASTER.CPP	404637	0.656341	DEO20.CPP	364915	0.494505	16835182	87

Figure 2.7 establishes a graphical relationship between the encryption time and decryption time for these *.CPP* files to draw the conclusion that here also there exists an almost linear relationship between these two values.

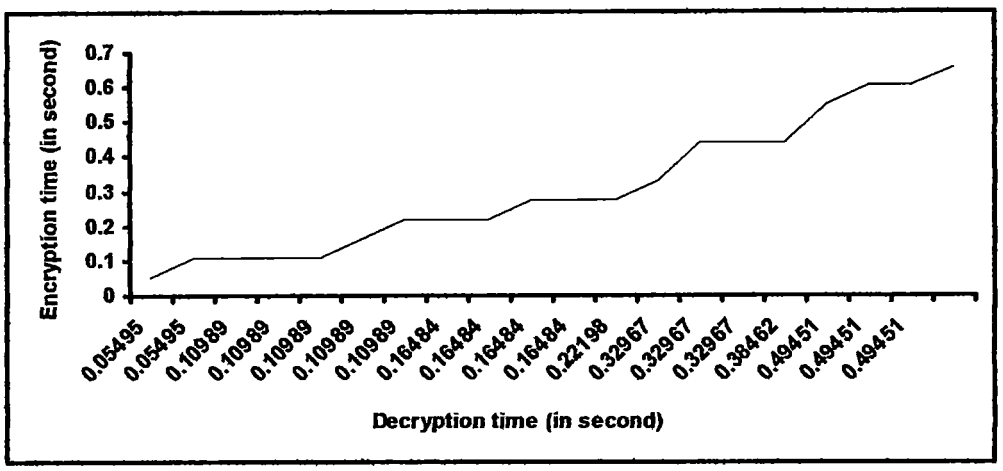

Figure 2.7
Graph to establish relationship between encryption time and decryption time for *.CPP* files for RCA technique

2.4.1.6 Discussion on Chi square tests

As it is observed from the results of the Chi square tests taken for .EXE, .DOC, .DLL, .SYS and .CPP files, Chi square values for .DOC are more in comparison to values for other files.

Figure 2.8 shows how Chi square values change with source file size only for the category of .CPP files. From this figure any fixed conclusion hardly can be drawn. As the Chi square value actually depends on the content of the source and the corresponding encrypted file, for files of almost same size Chi square values may differ to a large extend. But it is observed that there exists a tendency that the Chi square value increases with the file size, although the rate at which it increases is certainly not fixed and there exist a number of exceptions too.

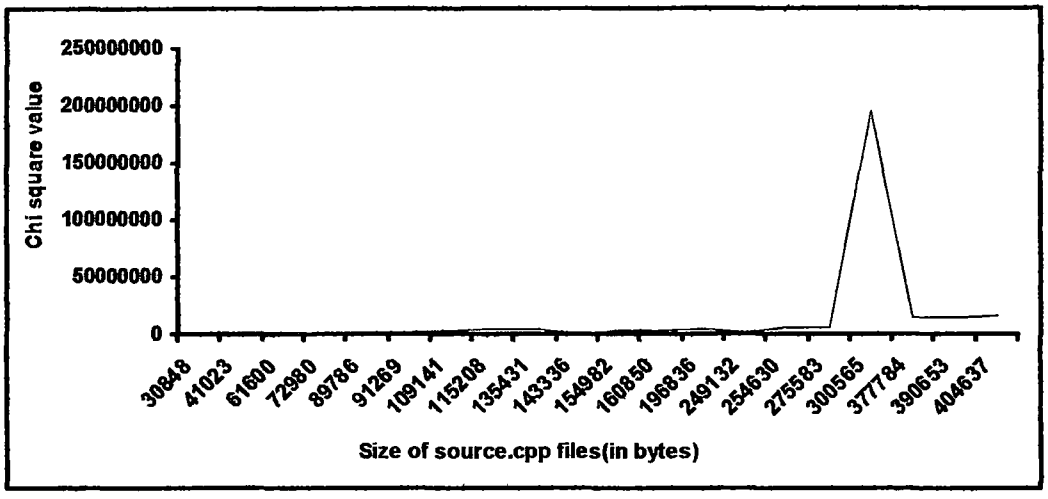

Figure 2.8
Variation of Chi square values with file sizes for .CPP files for RCA technique

Figure 2.9 attempts to represent a graphical outlook of a comparative analysis of the averages Chi square values and the source file size for all the five categories considered here. These average values are enlisted in table 2.18. From the table and the figure, it is observed that the result is most satisfactory in case of the .DOC and .CPP files, where the average value is found to be 25808915 and 1423338, in comparison with 8298102 for .DLL files, 2510091 for .SYS files, and 3233954 for .EXE files.

Table 2.18
Average Chi square values for different categories of files for RCA technique

Category of files	Average of Chi square value
.EXE	3233954
.DOC	25808915
.DLL	8298102
.SYS	2510091
.CPP	14233384

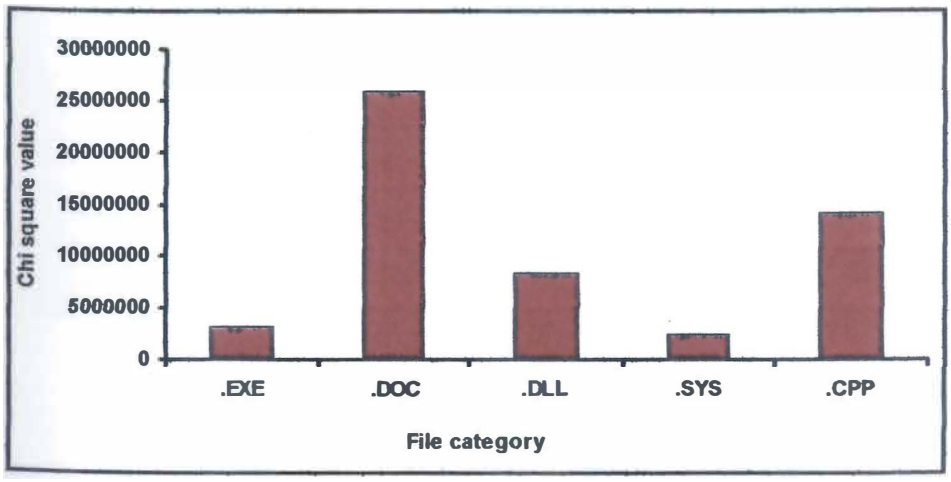

Figure 2.9
Graphical representation of average Chi square value for different categories of files for RCA technique

2.4.2 Result for frequency distribution tests

The frequency of each of the 255 characters in the source file and the same in the encrypted file were calculated and compared to assess the efficiency of the proposed technique. For representation purpose only 5 files one from each .EXE, .DOC, .DLL, .SYS and .CPP files have been considered. In each case, frequency distribution is pictorially represented for the source file and the encrypted file. Figure 2.10 to figure 2.14 show segments of graphical outcome for the source file and encrypted file.

These figures show the frequency of characters in a message and that frequency of characters in the encrypted message for RCA technique. From these figures it is very much clear that the source and the corresponding encrypted files are heterogeneous in nature. This may be interpreted that the proposed technique obtains a good quality of encryption.

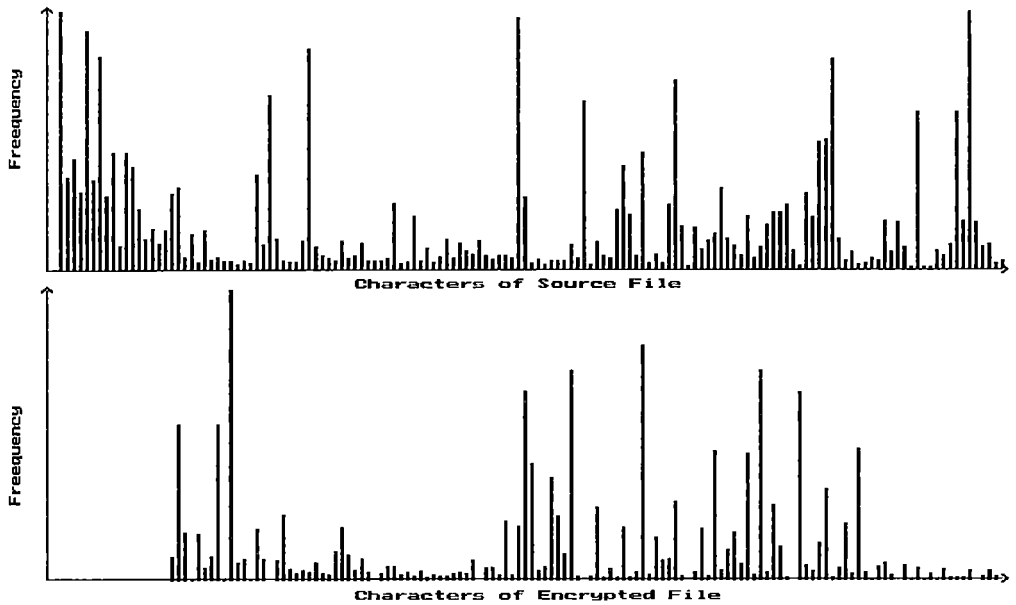

Figure 2.10
Frequency distribution of characters for source file TLIB.EXE and corresponding encrypted file DEO1.EXE for RCA technique

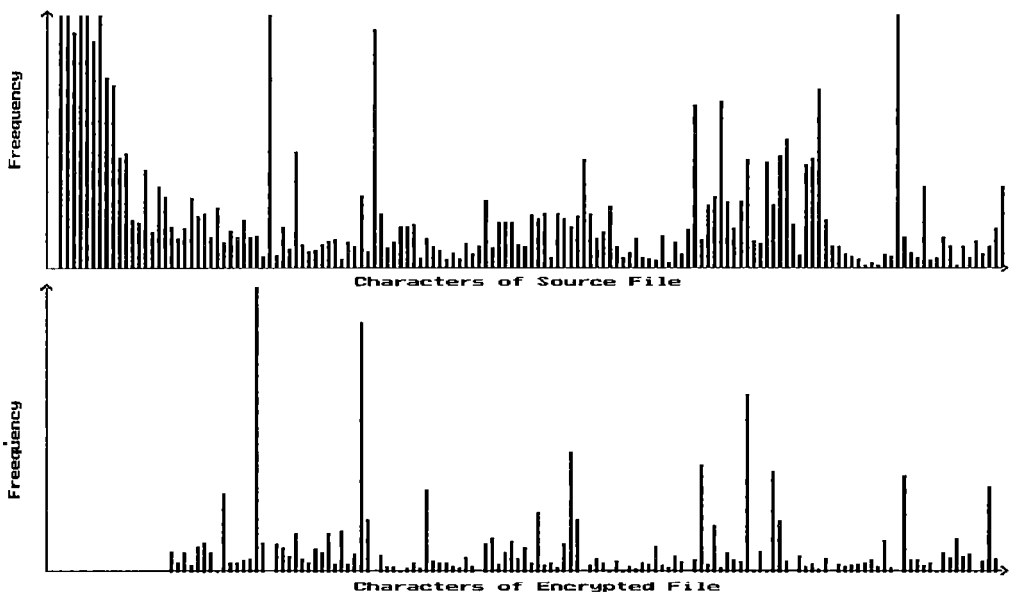

Figure 2.11
Frequency distribution of characters for source file QUESTION.DOC and corresponding encrypted file DEO1.DOC for RCA technique

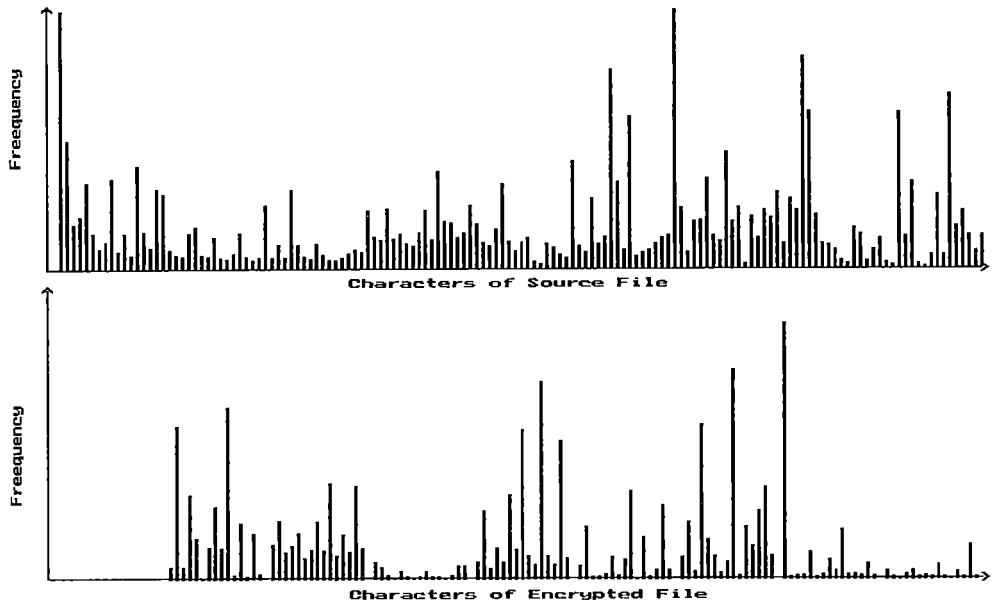

Figure 2.12
Frequency distribution of characters for source file ENUMVAR.dll and corresponding encrypted file DEO1.dll for RCA technique

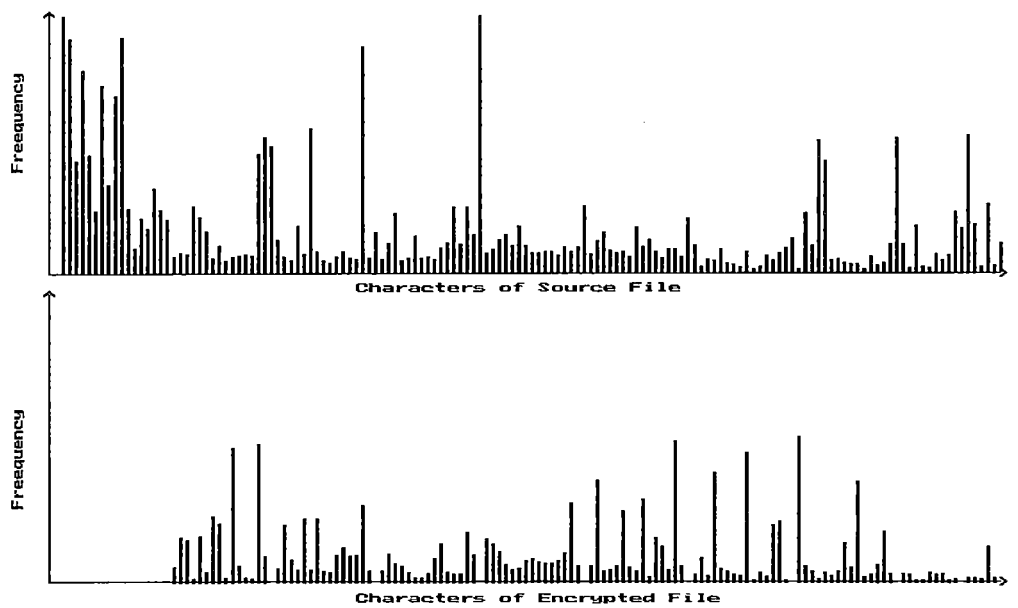

Figure 2.13
Frequency distribution of characters for source file NTIO.SYS and corresponding encrypted file DEO1.SYS for RCA technique

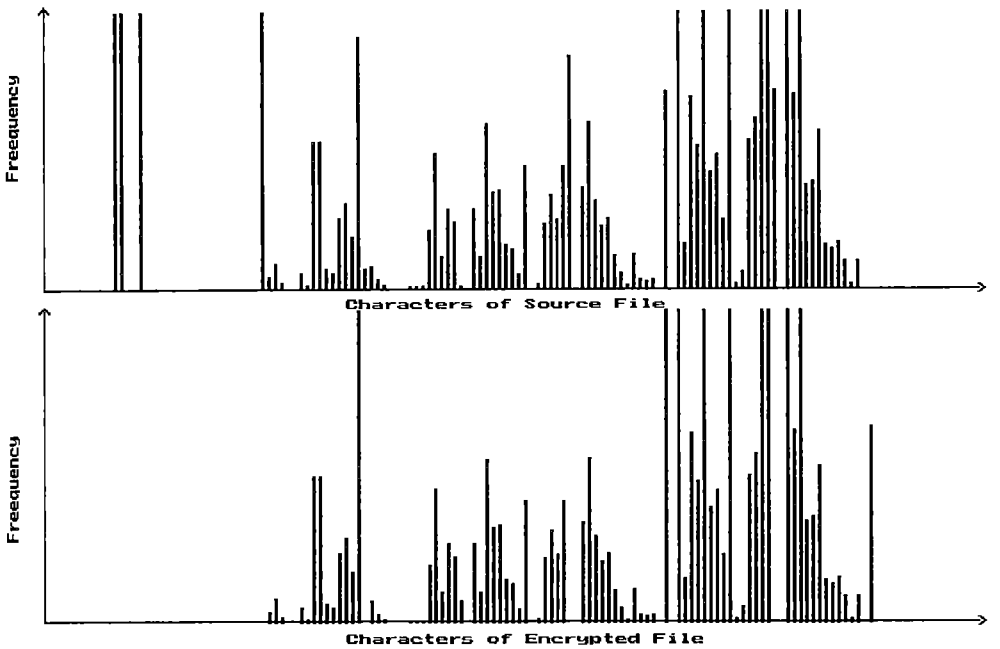

Figure 2.14
Frequency distribution of characters for source file VIEWPREV.CPP and corresponding encrypted file DEO1.CPP for RCA technique

2.4.3 Comparison with RSA and TDES technique

Section 2.4.3.1 presents the comparison among the RCA, RSA and TDES technique in terms of Chi square value and section 2.4.3.2 presents the comparison among the RCA, RSA and TDES technique in terms of frequency distribution for different categories of files, .EXE, .DOC, .DLL, .SYS, and .CPP respectively.

2.4.3.1 Comparison of RCA with RSA and TDES in terms of Chi square value

Section 2.4.3.1.1 to section 2.4.3.1.5 presents the comparison among RCA, RSA and TDES technique in terms of Chi square value.

2.4.3.1.1 Comparison of RCA with RSA and TDES for .EXE files

The result of implementing the proposed RCA technique on the sample .EXE files has been compared with the result of implementing the existing RSA and TDES technique on the same set of files. The comparison is made in terms of Chi square values. Table 2.19 enlists this comparative performance. Here the same twenty files of type .EXE have been considered, sizes of which are ranging from 32033 bytes to 4325428 bytes. The Chi square value between a source file and the corresponding encrypted file,

encrypted using the proposed RCA technique, is in the range of 126567 to 8257955, whereas the same between a source file and the corresponding encrypted file, encrypted using the existing RSA technique, is in the range of 3098 to 80485546 and the same between a source file and the corresponding encrypted file, encrypted using the existing TDES technique, is in the range of 4287 to 59350930. The degrees of freedom are in the range of 253 to 256.

Table 2.19
Comparison of Chi square values of RCA and RSA and TDES technique for .EXE files

Source file	Encrypted files using			Chi square value			Degree of freedom		
	RCA	RSA	TDES	RCA	RSA	TDES	RCA	RSA	TDES
TLIB.EXE	DEO1.EXE	RSE1.EXE	DES1.EXE	300224	54598	49476	255	253	256
PCLINK.EXE	DEO2.EXE	RSE2.EXE	DES2.EXE	126567	82011	74238	255	255	256
TEMC.EXE	DEO3.EXE	RSE3.EXE	DES3.EXE	370144	3098	4287	255	255	256
BRCC32.EXE	DEO4.EXE	RSE4.EXE	DES4.EXE	929417	104968	278570	255	256	256
WAVOTOASF.EXE	DEO5.EXE	RSE5.EXE	DES5.EXE	981575	1298326	96438	255	256	256
EXTRACT.EXE	DEO6.EXE	RSE6.EXE	DES6.EXE	511094	241720	91313	255	256	256
KRNL386.EXE	DEO7.EXE	RSE7.EXE	DES7.EXE	615273	622970	1155929	255	256	256
UPDATE.EXE	DEO8.EXE	RSE8.EXE	DES8.EXE	482329	4201613	533935	255	256	256
UTLIDX.EXE	DEO9.EXE	RSE9.EXE	DES9.EXE	880021	1656395	1959737	255	256	256
LINK.EXE	DEO10.EXE	RSE10.EXE	DES10.EXE	5908235	2885553	27992048	255	256	256
AUTOCONV.EXE	DEO11.EXE	RSE11.EXE	DES11.EXE	7233528	10272614	59350930	255	256	256
TD.EXE	DEO12.EXE	RSE12.EXE	DES12.EXE	5295981	15293932	18969329	255	256	256
TC.EXE	DEO13.EXE	RSE13.EXE	DES13.EXE	4548804	18723105	41695436	255	256	256
WINZIP32.EXE	DEO14.EXE	RSE14.EXE	DES14.EXE	3226002	17732878	21283453	255	256	256
BCW.EXE	DEO15.EXE	RSE15.EXE	DES15.EXE	6501097	25825051	19922757	255	256	256
TT6.EXE	DEO16.EXE	RSE16.EXE	DES16.EXE	3580608	17639838	49199137	255	256	256
WWF4.EXE	DEO17.EXE	RSE17.EXE	DES17.EXE	2349555	14556227	15217015	255	254	256
MIDTOWN.EXE	DEO18.EXE	RSE18.EXE	DES18.EXE	5621612	21037657	10234266	255	255	256
SWISH200.EXE	DEO19.EXE	RSE19.EXE	DES19.EXE	8257955	80485546	19884872	255	252	256
POWERPNT.EXE	DEO20.EXE	RSE20.EXE	DES20.EXE	6959065	16120368	12351657	255	254	256

Graphically, using horizontal bars, it has been exhibited in figure 2.15. Each purple bar stands for the Chi square value obtained implementing the RCA technique, and each red bar stands for the Chi square value obtained implementing the RSA technique and black bar stands for the TDES technique. It is observed from the graph and the table that in first almost six to seven cases the Chi square values is high in

comparison to the RSA technique and rest of the files the Chi square value for RSA is higher than RCA technique. Similarly when compared with the TDES technique only six files shows better result when encrypted with RCA technique and for the rest files it shows comparable result.

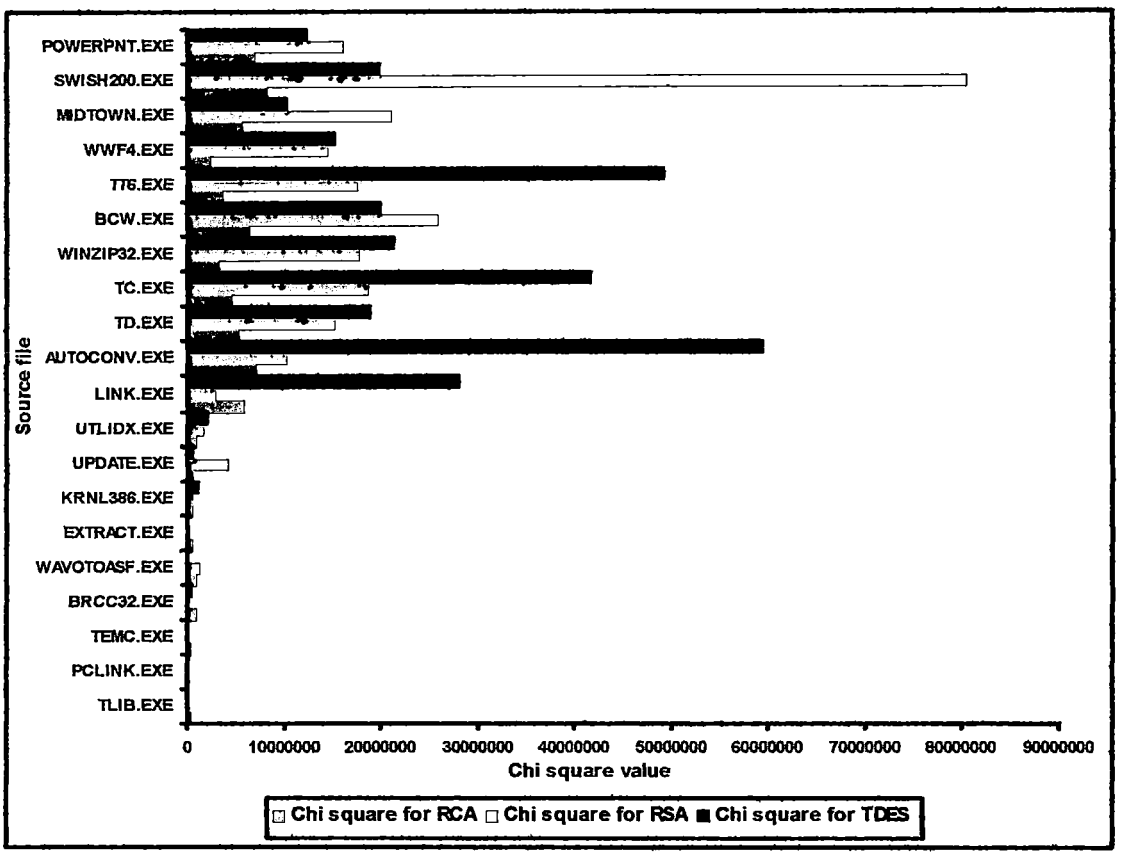

Figure 2.15
Graphical comparison of Chi square values for RCA, RSA and TDES technique for .EXE file

2.4.3.1.2 Comparison of RCA with RSA and TDES for .DOC files

The result of implementing the proposed RCA technique on the sample .DOC files has been compared with the result of implementing the existing RSA and TDES technique on the same set of files. The comparison is made in terms of Chi square values. Table 2.20 enlists this comparative performance. Here the same twenty files of type .DOC have been considered, sizes of which are ranging from 106496 bytes to 4741120 bytes. The Chi square value between a source file and the corresponding encrypted file, encrypted using the proposed RCA technique, is in the range of 61669 to 372610016,

whereas the same between a source file and the corresponding encrypted file, encrypted using the existing RSA technique, is in the range of 1084638 to 80668188 and same between a source file and the corresponding encrypted file, encrypted using the existing TDES technique, is in the range of 1047479 to 94308642. The degrees of freedom are in the range of 115 to 256.

Table 2.20
Comparison of Chi square values of RCA, RSA and TDES technique for .DOC files

Source file	Encrypted files using			Chi square value			Degree of freedom		
	RCA	RSA	TDES	RCA	RSA	TDES	RCA	RSA	TDES
QUESTION.DOC	DEO1.DOC	RSE1.DOC	DES1.DOC	273599	5136590	1047479	224	254	256
RIANGLE.DOC	DEO2.DOC	RSE2.DOC	DES2.DOC	225896	2914601	1515581	224	255	256
LINUX.DOC	DEO3.DOC	RSE3.DOC	DES3.DOC	632584	2460712	22407084	224	256	256
UTIL.DOC	DEO4.DOC	RSE4.DOC	DES4.DOC	12380527	6877968	7408074	224	119	256
CLASSLIB.DOC	DEO5.DOC	RSE5.DOC	DES5.DOC	30681776	4604245	8025788	224	137	256
UNIX6.DOC	DEO6.DOC	RSE6.DOC	DES6.DOC	12619891	1084638	20292913	224	256	256
PROCESS1.DOC	DEO7.DOC	RSE7.DOC	DES7.DOC	171720	8949619	45418958	224	256	256
UNIX1.DOC	DEO8.DOC	RSE8.DOC	DES8.DOC	4144433	4743222	82638502	224	256	256
THIRTEEN.DOC	DEO9.DOC	RSE9.DOC	DES9.DOC	77930	8949619	76470343	224	256	256
FINAL.DOC	DEO10.DOC	RSE10.DOC	DES10.DOC	8535466	9862052	49591157	224	255	256
TWELVE.DOC	DEO11.DOC	RSE11.DOC	DES11.DOC	61669	53180200	10877345	224	256	255
HISTORY.DOC	DEO12.DOC	RSE12.DOC	DES12.DOC	444870	16160158	34262048	224	256	256
DEBUGREF.DOC	DEO13.DOC	RSE13.DOC	DES13.DOC	7465953	9879969	11326775	224	256	254
ELEVEN.DOC	DEO14.DOC	RSE14.DOC	DES14.DOC	94041	17325131	94308642	224	255	253
COURSE.DOC	DEO15.DOC	RSE15.DOC	DES15.DOC	5960380	14399585	55674733	224	255	253
PART.DOC	DEO16.DOC	RSE16.DOC	DES16.DOC	55526240	80668188	60115974	224	118	254
PEARL.DOC	DEO17.DOC	RSE17.DOC	DES17.DOC	372610016	18392785	83933540	224	115	252
DOTNET.DOC	DEO18.DOC	RSE18.DOC	DES18.DOC	3096224	11486702	17267936	224	255	252
MEMORY.DOC	DEO19.DOC	RSE19.DOC	DES19.DOC	205144	20823688	18721296	224	251	254
FILEMMT.DOC	DEO20.DOC	RSE20.DOC	DES20.DOC	969945	35971113	19058308	224	253	254

Graphically, using horizontal bars, it has been exhibited in figure 2.16. Each purple bar stands for the Chi square value obtained implementing the RCA technique, each red bar stands for the Chi square value obtained implementing the RSA technique and the black bar stands for the TDES technique. It is observed from the graph and the table that the Chi square value for RCA technique shows comparable result when compared with the RSA and TDES technique.

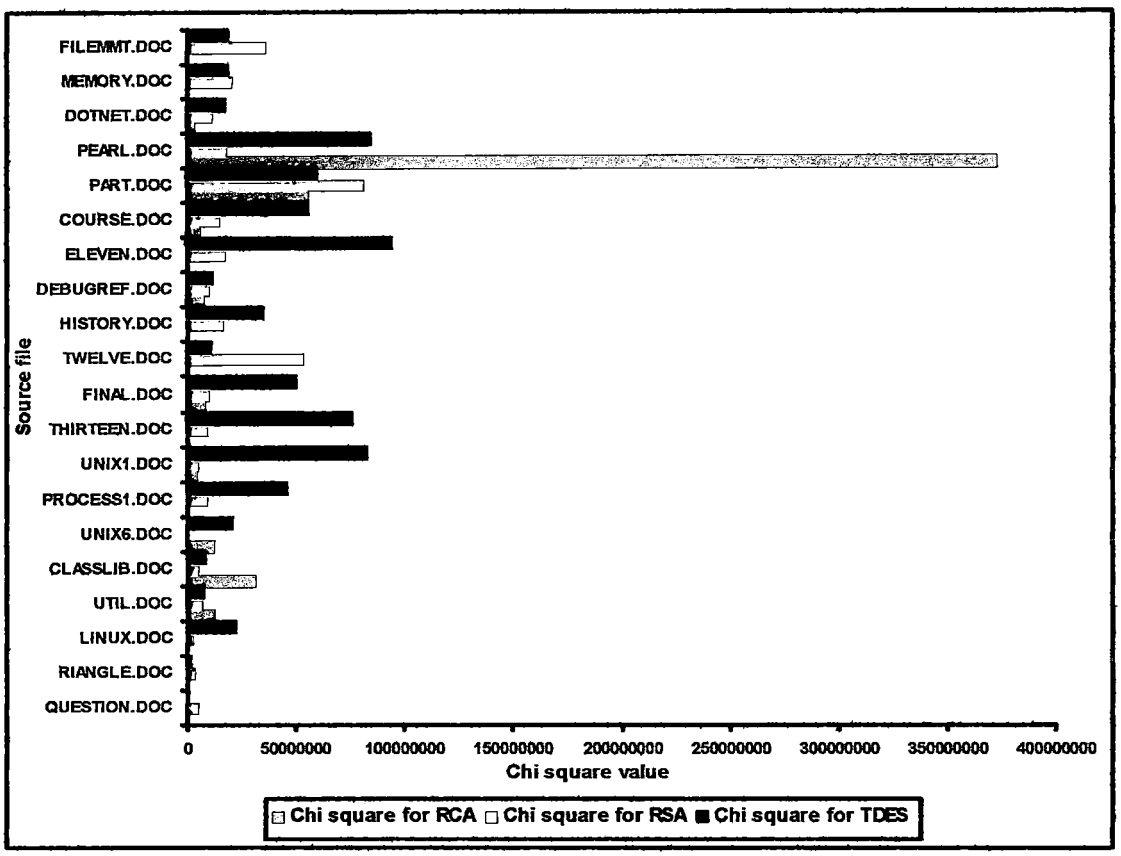

Figure 2.16
Graphical comparison of Chi square values for RCA, RSA and TDES technique for .DOC file

2.4.3.1.3 Comparison of RCA with RSA and TDES for .DLL files

The result of implementing the proposed RCA technique on the sample .DLL files has been compared with the result of implementing the existing RSA and TDES technique on the same set of files. The comparison is made in terms of Chi square values. Table 2.21 enlists this comparative performance. Here the same twenty files of type .DLL have been considered, sizes of which are ranging from 28685 bytes to 5582897 bytes. The Chi square value between a source file and the corresponding encrypted file, encrypted using the proposed RCA technique, is in the range of 102783 to 55551000, whereas the same between a source file and the corresponding encrypted file, encrypted using the existing RSA technique, is in the range of 49725 to 77673838 and the same between a source file and the corresponding encrypted file, encrypted using the existing

TDES technique, is in the range of 51113 to 95080869 and degree of freedom, is in the range of 191 to 256.

Table 2.21
Comparison of Chi square values of RCA, RSA and TDES technique for .DLL files

Source file	Encrypted files using			Chi square value			Degree of freedom		
	RCA	RSA	TDES	RCA	RSA	TDES	RCA	RSA	TDES
ENUMVAR.DLL	DEO1.DLL	RSE1.DLL	DES1.DLL	347732	49725	53733	224	201	256
SELFREG.DLL	DEO2.DLL	RSE2.DLL	DES2.DLL	102783	55460	51113	219	251	256
VCARD.DLL	DEO3.DLL	RSE3.DLL	DES3.DLL	468712	62578	577607	224	256	256
ACTPANEL.DLL	DEO4.DLL	RSE4.DLL	DES4.DLL	266242	90937	84458	224	255	256
WZCAB.DLL	DEO5.DLL	RSE5.DLL	DES5.DLL	764611	112704	102727	224	255	256
ODBCINT.DLL	DEO6.DLL	RSE6.DLL	DES6.DLL	4210088	523023	2563201	206	191	256
VSAMI.DLL	DEO7.DLL	RSE7.DLL	DES7.DLL	827770	348676	927682	224	256	256
MFC42.DLL	DEO8.DLL	RSE8.DLL	DES8.DLL	8989041	2058606	95080869	224	255	254
OWL52.DLL	DEO9.DLL	RSE9.DLL	DES9.DLL	8273230	10261680	14749250	224	256	255
BFC40.DLL	DEO10DLL	RSE10.DLL	DES10.DLL	9272143	77673838	27368209	224	256	256
DEVSHL.DLL	DEO11.DLL	RSE11.DLL	DES11.DLL	8447326	35212506	47076034	224	256	255
COOLTYPE.DLL	DEO12.DLL	RSE12.DLL	DES12.DLL	7228557	19077332	18314381	224	256	251
VGX.DLL	DEO13.DLL	RSE13.DLL	DES13.DLL	3525923	17393607	33275056	224	253	254
ENGINE.DLL	DEO14.DLL	RSE14.DLL	DES14.DLL	7060996	13098959	43330787	224	253	252
JVM.DLL	DEO15.DLL	RSE15.DLL	DES15.DLL	9754395	20100180	85886103	224	253	253
MSENV.DLL	DEO16.DLL	RSE16.DLL	DES16.DLL	6221269	12499308	20578502	224	256	254
MSO97RT.DLL	DEO17.DLL	RSE17.DLL	DES17.DLL	2595513	19126073	88456420	224	254	252
JVM_G.DLL	DEO18.DLL	RSE18.DLL	DES18.DLL	20628350	12237396	13898511	224	251	252
OUTLLIB.DLL	DEO19.DLL	RSE19.DLL	DES19.DLL	11426359	15855153	39029212	224	253	254
MSO9.DLL	DEO20.DLL	RSE20.DLL	DES20.DLL	55551000	16393094	16319404	224	247	254

Graphically, using horizontal bars, it has been exhibited in figure 2.17. Each purple bar stands for the Chi square value obtained implementing the RCA technique, and each red bar stands for the Chi square value obtained implementing the RSA technique and each black bar stands for the TDES technique. It is observed from the graph and the table that in almost eight cases the Chi square value is high in comparison to the RSA technique and rest of the cases Chi square value of RSA is higher than the RCA technique. Similarly when it is compared with TDES technique it may be deduced that the Chi square value of RCA technique is comparable with the TDES technique.

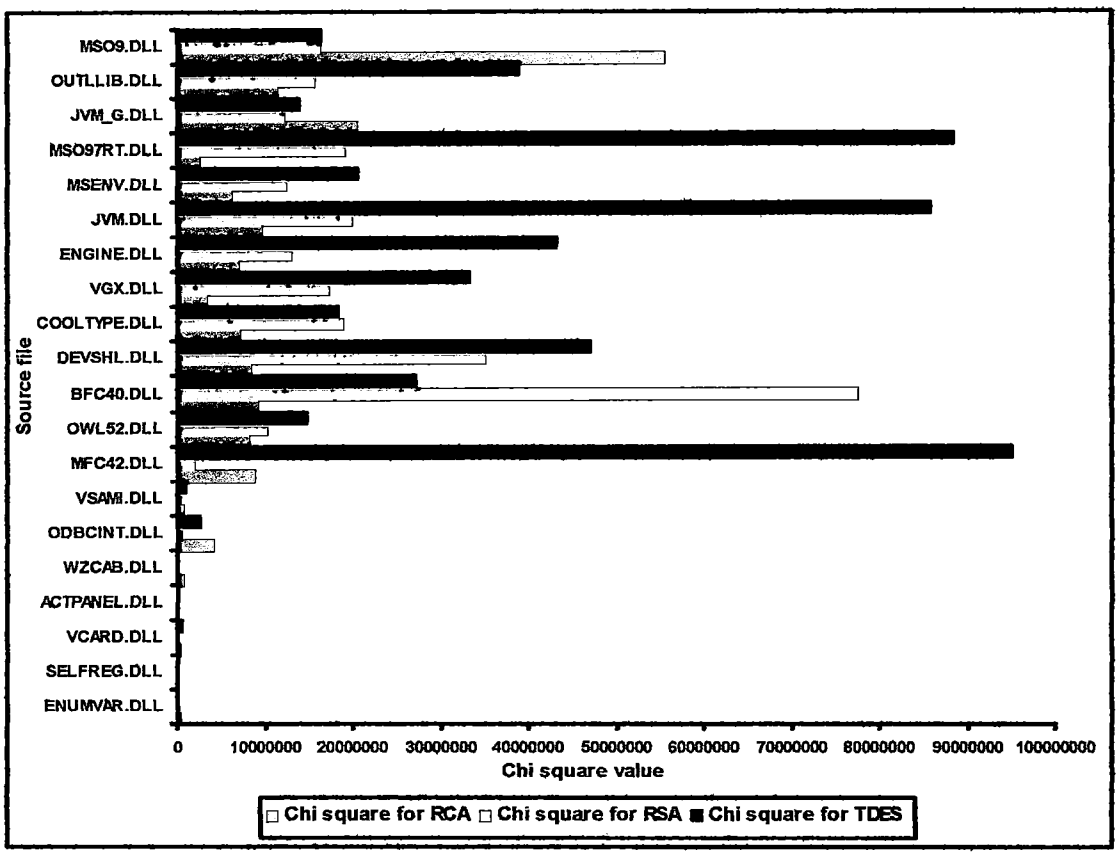

Figure 2.17
Graphical comparison of Chi square values for RCA, RSA and TDES technique for .DLL file

2.4.3.1.4 Comparison of RCA with RSA and TDES for .SYS files

The result of implementing the proposed RCA technique on the sample .SYS files has been compared with the result of implementing the existing RSA and the TDES technique on the same set of files. The comparison is made in terms of Chi square values. Table 2.22 enlists this comparative performance. Here the same twenty files of type .SYS have been considered, sizes of which are ranging from 35392 bytes to 1748399 bytes. The Chi square value between a source file and the corresponding encrypted file, encrypted using the proposed RCA technique, is in the range of 114958 to 8694071, whereas the same between a source file and the corresponding encrypted file, encrypted using the existing RSA technique, is in the range of 58878 to 67935312 and the same between a source file and the corresponding encrypted file, encrypted using the existing

TDES technique, is in the range of 51434 to 80582596. The degrees of freedom are in the range of 252 to 256.

Table 2.22
Comparison of Chi square values of RCA, RSA and TDES technique for .SYS files

Source file	Encrypted files using			Chi square value			Degree of freedom		
	RCA	RSA	TDES	RCA	RSA	TDES	RCA	RSA	TDES
NTIO412.SYS	DEO1.SYS	RSE1.SYS	DES1.SYS	114958	58878	54384	254	255	256
I8O42PRT.SYS	DEO2.SYS	RSE2.SYS	DES2.SYS	309313	76136	70201	254	256	256
DLC.SYS	DEO3.SYS	RSE3.SYS	DES3.SYS	304224	94192	85520	254	256	256
NAVENG.SYS	DEO4.SYS	RSE4.SYS	DES4.SYS	680246	108511	101452	254	256	256
LVCODEK.SYS	DEO5.SYS	RSE5.SYS	DES5.SYS	239757	141036	22058	254	256	256
NBF.SYS	DEO6.SYS	RSE6.SYS	DES6.SYS	938112	276188	51434	254	256	256
DMIO.SYS	DEO7.SYS	RSE7.SYS	DES7.SYS	1137828	178737	1457544	254	256	256
NETBT.SYS	DEO8.SYS	RSE8.SYS	DES8.SYS	1773381	133220	239674	254	256	256
NDIS.SYS	DEO9.SYS	RSE9.SYS	DES9.SYS	1391272	3351319	418199	254	256	256
SRV.SYS	DEO10.SYS	RSE10.SYS	DES10.SYS	1508606	78319	1244811	254	256	256
TCPIP.SYS	DEO11.SYS	RSE11.SYS	DES11.SYS	644123	14852419	2835045	254	256	256
DMBOOT.SYS	DEO12.SYS	RSE12.SYS	DES12.SYS	2272823	63424175	13246072	254	256	256
NAVEXI5.SYS	DEO13.SYS	RSE13.SYS	DES13.SYS	2823472	8951599	2814749	254	256	256
NTFS.SYS	DEO14.SYS	RSE14.SYS	DES14.SYS	2620915	11984076	31208845	254	256	256
VMODEM.SYS	DEO15.SYS	RSE15.SYS	DES15.SYS	4136534	38420327	15511384	254	256	256
DEMO.SYS	DEO16.SYS	RSE16.SYS	DES16.SYS	7963321	13035095	21956219	254	256	256
VPCTCOM.SYS	DEO17.SYS	RSE17.SYS	DES17.SYS	1078467	67935312	31232004	254	255	255
DISTROY.SYS	DEO18.SYS	RSE18.SYS	DES18.SYS	2816246	23535178	11397406	254	255	256
WIN32K.SYS	DEO19.SYS	RSE19.SYS	DES19.SYS	8754148	18810298	80582596	254	255	256
DIRECT.SYS	DEO20.SYS	RSE20.SYS	DES20.SYS	8694071	33802717	16493549	254	255	252

Graphically, using horizontal bars, it has been exhibited in figure 2.17. Each purple bar stands for the Chi square value obtained implementing the RCA technique, and each red bar stands for the Chi square value obtained implementing the RSA technique and each black bar stands for the TDES technique. It is observed from the graph and the table that in almost eight cases the Chi square value is high in comparison to the RSA technique and rest of the cases Chi square value of RSA is higher than the RCA technique and when it is compared with the TDES technique it shows comparable result.

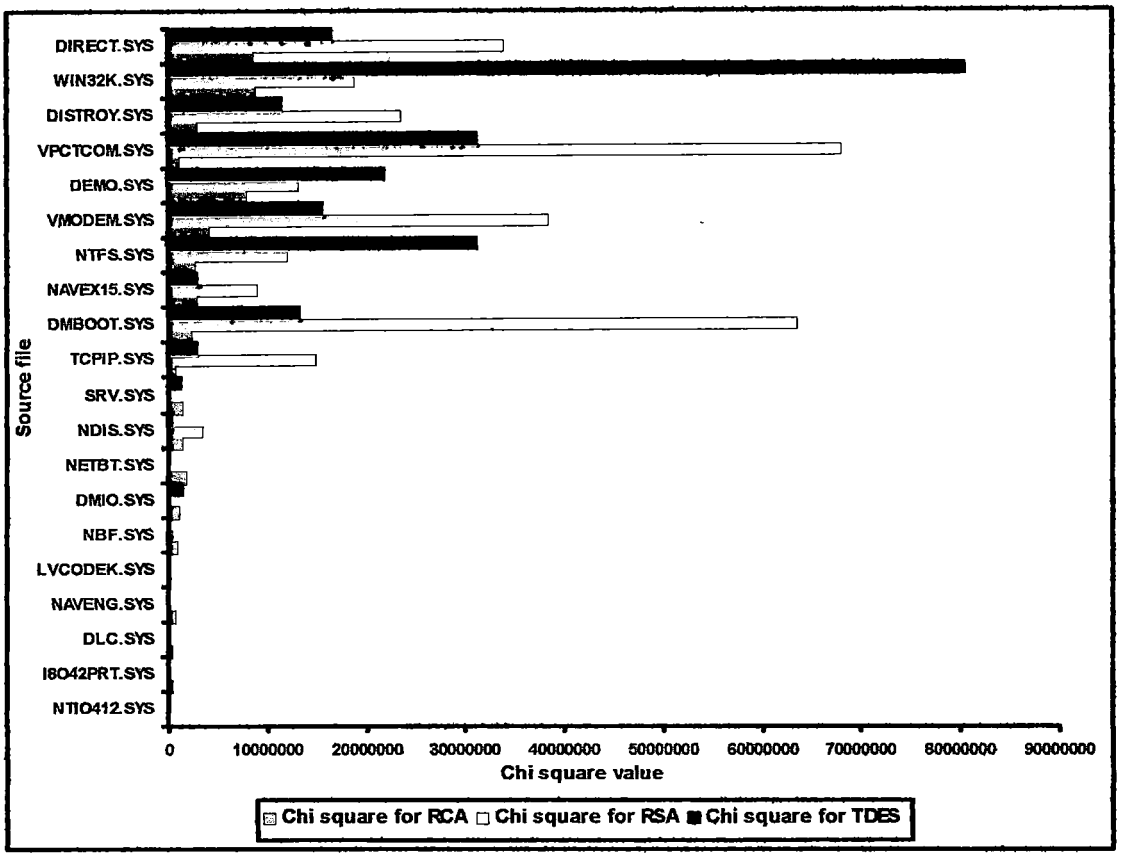

Figure 2.18
Graphical comparison of Chi square values for RCA, RSA and TDES technique for .SYS file

2.4.3.1.5 Comparison of RCA with RSA and TDES for .CPP files

The result of implementing the proposed RCA technique on the sample .CPP files has been compared with the result of implementing the existing RSA and the TDES technique on the same set of files. Table 2.23 enlists this comparative performance. Here the same twenty files of type .CPP have been considered, sizes of which are ranging from 30848 bytes to 404637 bytes. The Chi square value between a source file and the corresponding encrypted file, encrypted using the proposed RCA technique, is in the range of 39447 to 194565600, whereas the same between a source file and the corresponding encrypted file, encrypted using the existing RSA technique, is in the range of 215853 to 24641444 and the same between a source file and the corresponding encrypted file, encrypted using the existing TDES technique, is in the range of 35672 to 42480951. The degrees of freedom are in the range of 77 to 256.

Table 2.23
Comparison of Chi square values of RCA, RSA and TDES technique for .CPP files

Source file	Encrypted files using			Chi square value			Degree of freedom		
	RCA	RSA	TDES	RCA	RSA	TDES	RCA	RSA	TDES
VIEWPREV.CPP	DEO1.SYS	RSE1.SYS	DES1.SYS	568921	1015121	54694	85	78	256
OLECLI2.CPP	DEO2.SYS	RSE2.SYS	DES2.SYS	39447	750711	73369	84	78	256
OLECLI1.CPP	DEO3.SYS	RSE3.SYS	DES3.SYS	59323	215853	109858	87	80	256
INET.CPP	DEO4.SYS	RSE4.SYS	DES4.SYS	1173146	223480	199820	86	80	256
OCCSITE.CPP	DEO5.SYS	RSE5.SYS	DES5.SYS	1561664	302856	159044	89	81	256
DBRFX.CPP	DEO6.SYS	RSE6.SYS	DES6.SYS	2044262	618369	35672	85	78	256
WINCORE.CPP	DEO7.SYS	RSE7.SYS	DES7.SYS	2659827	411302	194013	91	82	256
DBCORE.CPP	DEO8.SYS	RSE8.SYS	DES8.SYS	4291539	401363	204655	89	81	256
DAOCORE.CPP	DEO9.SYS	RSE9.SYS	DES9.SYS	5135549	380307	201857	89	81	256
BOOK.CPP	DEO10.SYS	RSE10.SYS	DES10.SYS	694662	19977116	269585	89	77	256
FLIGHT.CPP	DEO11.SYS	RSE11.SYS	DES11.SYS	2633964	708998	1445428	89	84	256
DRAMA.CPP	DEO12.SYS	RSE12.SYS	DES12.SYS	2633964	714752	1481626	90	84	256
GENIL.CPP	DEO13.SYS	RSE13.SYS	DES13.SYS	4400968	581653	1520052	90	81	256
DUIT.CPP	DEO14.SYS	RSE14.SYS	DES14.SYS	1491142	21316320	6291367	89	66	256
COMMANDS.CPP	DEO15.SYS	RSE15.SYS	DES15.SYS	7081902	9209219	5242928	80	83	256
PARSER.CPP	DEO16.SYS	RSE16.SYS	DES16.SYS	6854556	4068862	787244	90	82	256
DEAR.CPP	DEO17.SYS	RSE17.SYS	DES17.SYS	194565600	24641444	11226949	86	75	256
COMP.CPP	DEO18.SYS	RSE18.SYS	DES18.SYS	14971028	4332684	13367131	87	82	256
DOIT.CPP	DEO19.SYS	RSE19.SYS	DES19.SYS	14971028	4327673	20232918	87	82	256
MASTER.CPP	DEO20.SYS	RSE20.SYS	DES20.SYS	16835182	4870378	42480951	87	82	256

Graphically, using horizontal bars, it has been exhibited in figure 2.19. Each purple bar stands for the Chi square value obtained implementing the RCA technique, each red bar stands for the Chi square value obtained implementing the RSA technique and each black bar stands for the TDES technique. It is observed from the graph and the table that in almost all the cases the Chi square value is high in comparison to the RSA and TDES techniques. So RCA technique may be compared with the RSA and TDES techniques.

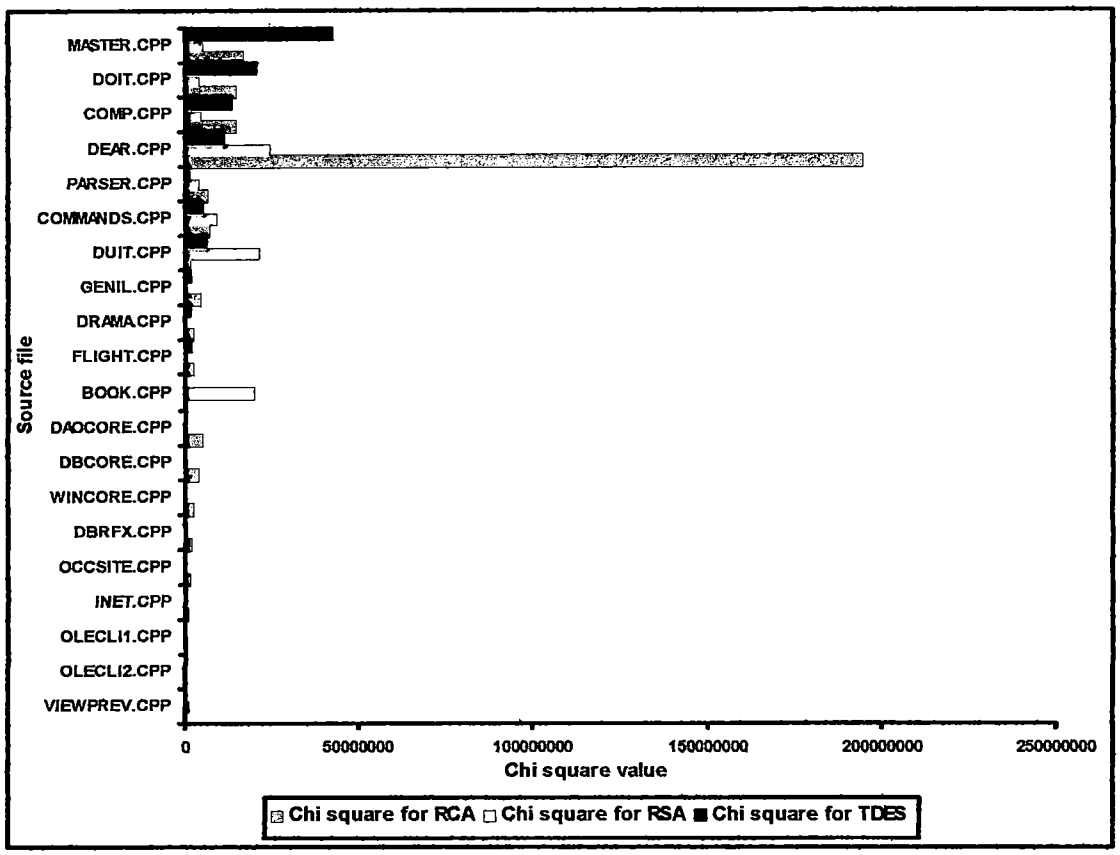

Figure 2.19
Graphical comparison of Chi square values for RCA, RSA and TDES technique for .CPP file

2.4.3.2 Comparison of RCA with RSA and TDES in terms of frequency distribution

Comparing the result of the frequency distribution test for all the files considered in section 2.4.1 being an impractical task, here in this section, for the comparison purpose only 5 files, one each from .EXE, .DOC, .DLL, .SYS, and .CPP have been considered. In each case, frequency distribution is pictorially represented for the source file and the encrypted file using RSA and TDES technique. When it is compared with the figure 2.20 to 2.24, it is seen in all cases the characters in the encrypted files are well distributed, which indicates that the technique proposed here is quite compatible with existing techniques. The frequencies of characters in encrypted files are evenly distributed. Therefore the source and the corresponding encrypted file are heterogeneous

in nature. Hence it can be interpreted that through the proposed technique, a good quality of encryption is obtained.

Section 2.4.3.2.1 to section 2.4.3.2.5 presents the comparison between RCA, RSA and TDES technique in terms of frequency distribution.

2.4.3.2.1 Comparison of RCA with RSA and TDES for .EXE files

Figure 2.20 shows the comparison of source file TLIB.EXE with RCA, RSA and TDES technique.

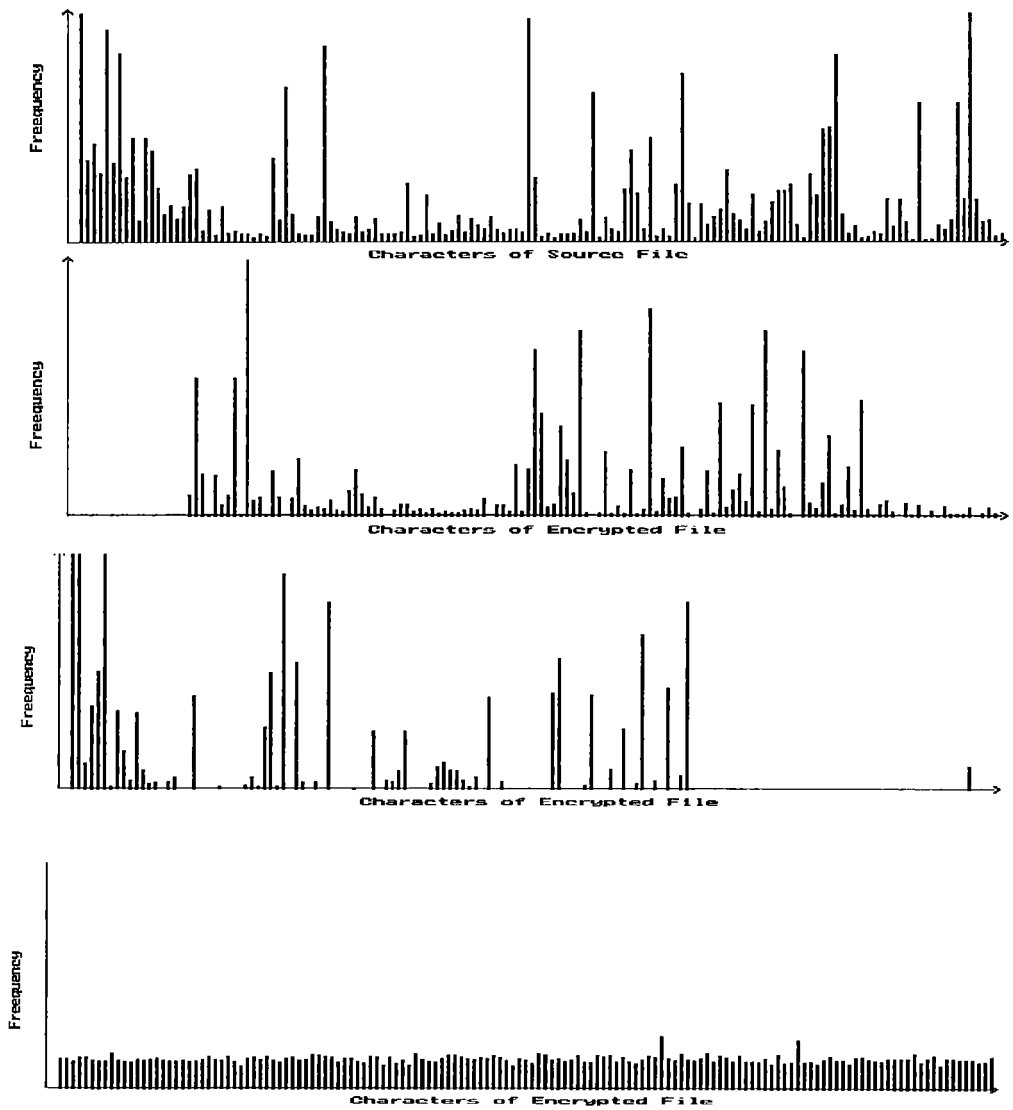

Figure 2.20
Frequency distribution for TLIB.EXE (source file) and its encrypted file (using RCA technique), encrypted file (using RSA technique) and encrypted file (using TDES technique).

2.4.3.2.2 Comparison of RCA with RSA and TDES for .DOC files

Figure 2.21 shows the comparison of source file QUESTION.DOC with RCA, RSA and TDES technique.

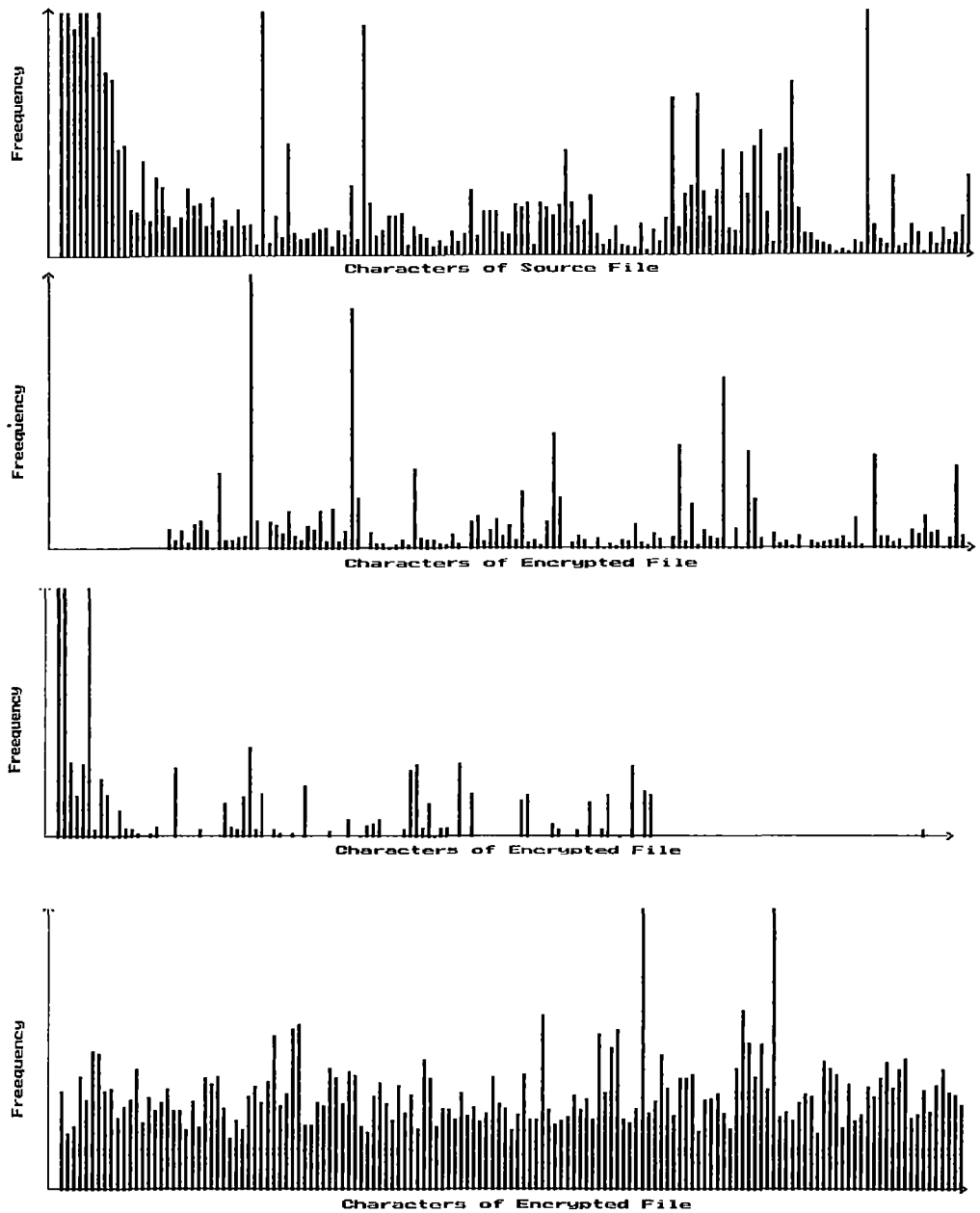

Figure 2.21

Frequency distribution for QUESTION.DOC (source file) and its encrypted file (using RCA technique), encrypted file (using RSA technique) and encrypted file (using TDES technique).

2.4.3.2.3 Comparison of RCA with RSA and TDES for .DLL files

Figure 2.22 shows the comparison of source file ENUMVAR.DLL with RCA, RSA and TDES technique.

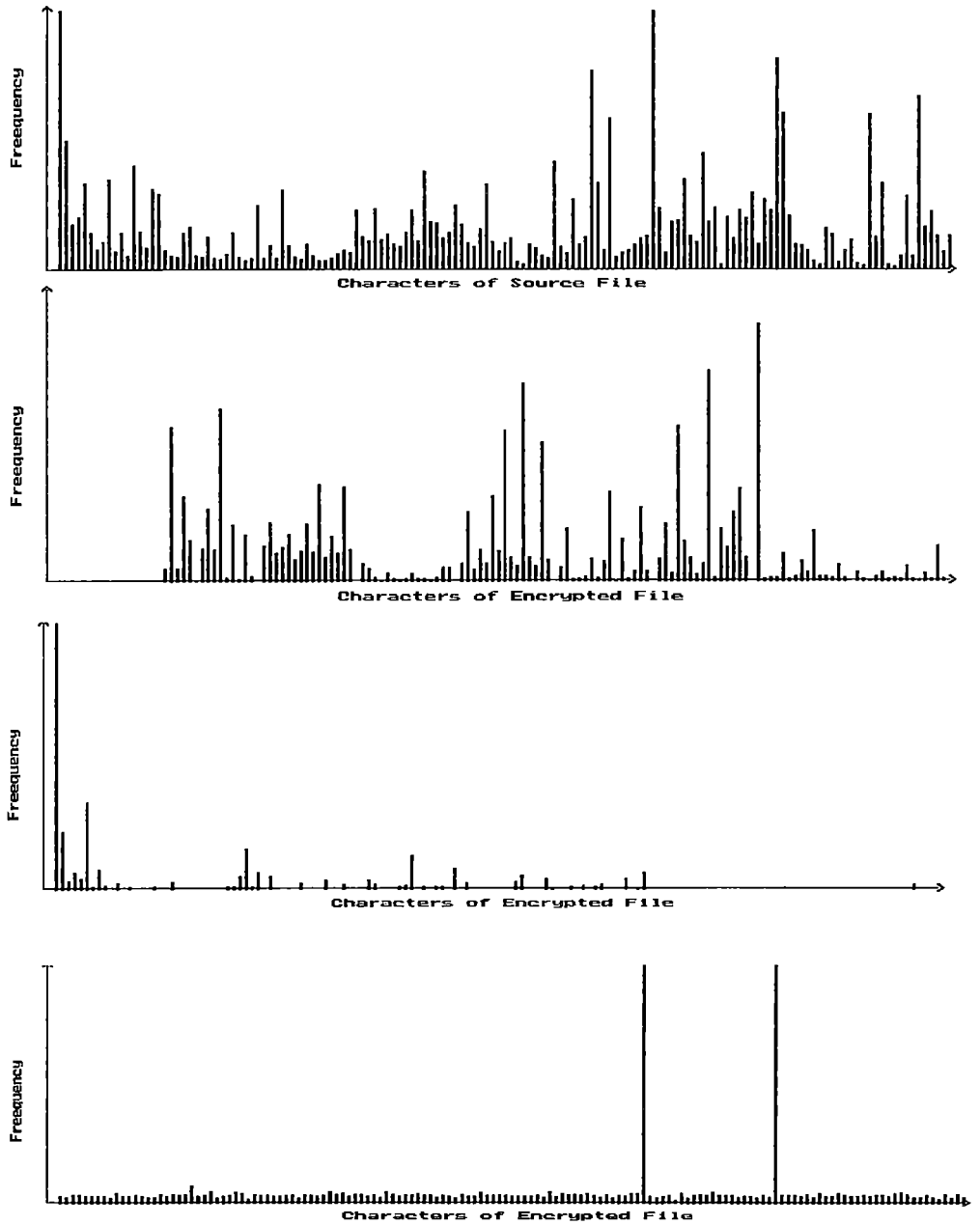

Figure 2.22
Frequency distribution for ENUMVAR.DLL (source file) and its encrypted file (using RCA technique), encrypted file (using RSA technique) and encrypted file (using TDES technique).

2.4.3.2.4 Comparison of RCA with RSA and TDES for .SYS files

Figure 2.23 shows the comparison of the source file NTIO412.SYS with RCA, RSA and TDES technique.

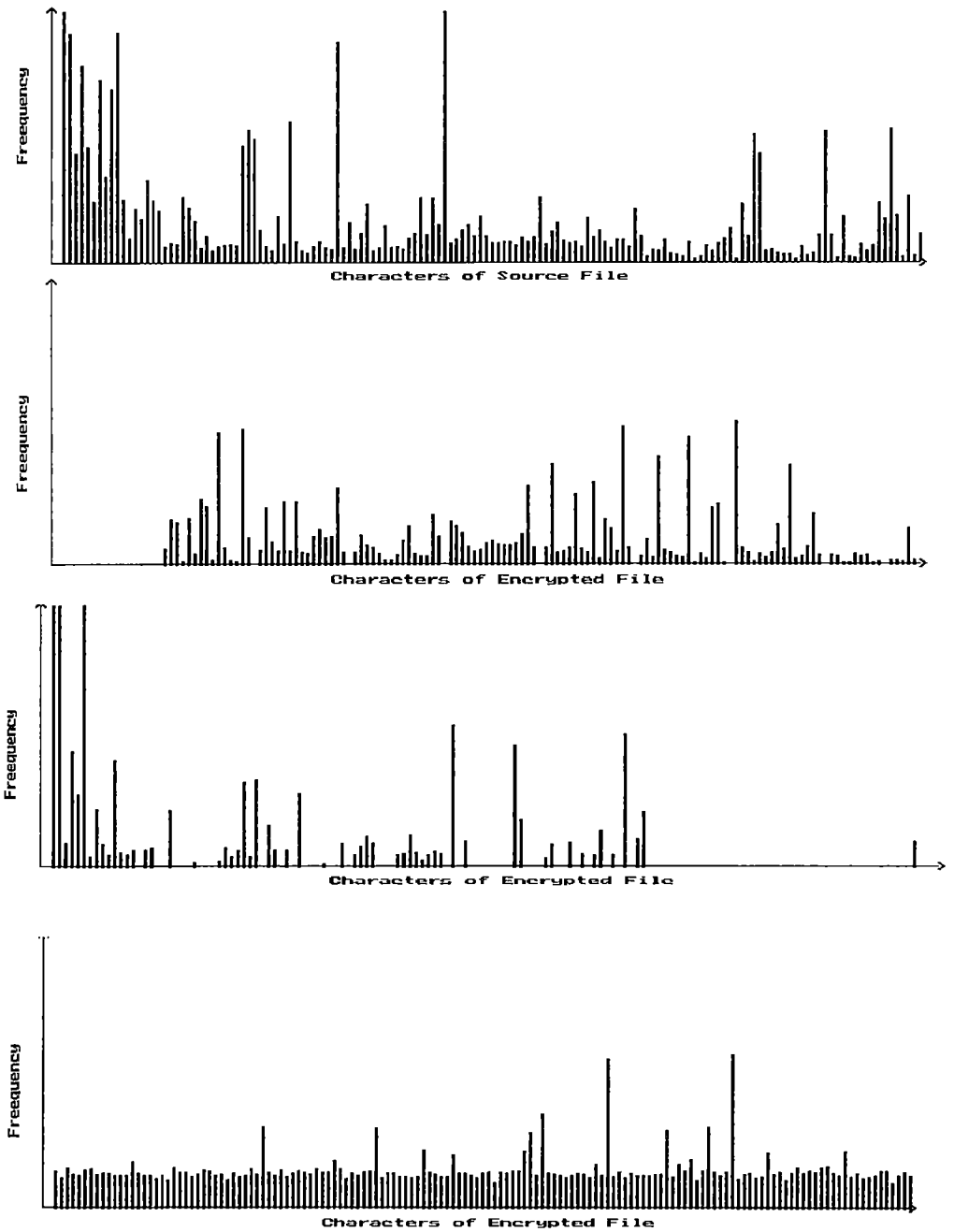

Figure 2.23
Frequency distribution for NTIO412.SYS (source file) and its encrypted file (using RCA technique), encrypted file (using RSA technique) and encrypted file (using TDES technique).

2.4.3.2.5 Comparison of RCA with RSA and TDES for .CPP files

Figure 2.24 shows the comparison of source file VIEWPREV.CPP with RCA, RSA and TDES technique.

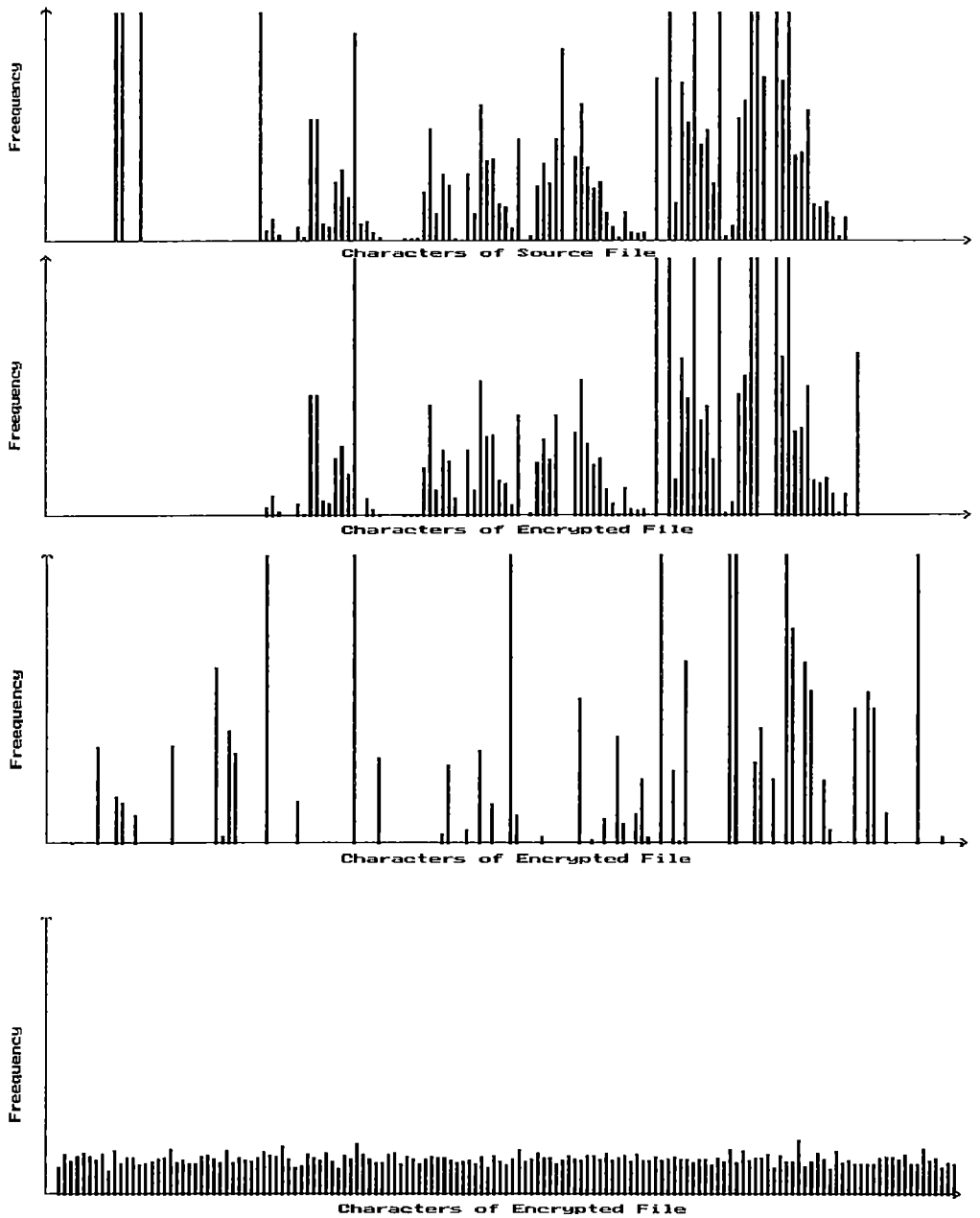

Figure 2.24
Frequency distribution for VIEWPREV.CPP (source file) and its encrypted file (using RCA technique), encrypted file (using RSA technique) and encrypted file (using TDES technique).

2.5 Analysis

This section consists of analysis on three issues. One is the issue related to block size, discussed in section 2.5.1. Another issue is related to different factors that were considered to evaluate the technique, discussed in section 2.5.2. The last issue is the key as well as vulnerability of the technique, which is analyzed in chapter 8, a special chapter on the key generation and vulnerability.

2.5.1 Analysis on block size

To enhance the security, one criterion should be to choose the block size such a manner that it requires a huge number of iterations to complete the cycle. Now, generally the number of such iterations increases as the block size increases. Table 2.24 shows how the number of such iterations changes with changes in block size.

Table 2.24
No. of iteration (I) required to form cycles for blocks of different sizes (N) for RCA technique

N	I
2	2
4	4
8	8
16	16
32	32
64	64
128	128
256	256

The graph shown in figure 2.25 to represent table 2.24 depicts the clear picture in this regard. In this graph, block sizes in the range from 2 to 256 have been considered. In that case it is observed that a steady increase in the number of iterations is required to complete the cycle, as the value of N increases.

Figure 2.25 shows that there exists a linear relationship between the number of iterations required to complete the cycle and the unique block size if the block size ranges from 2 to 256.

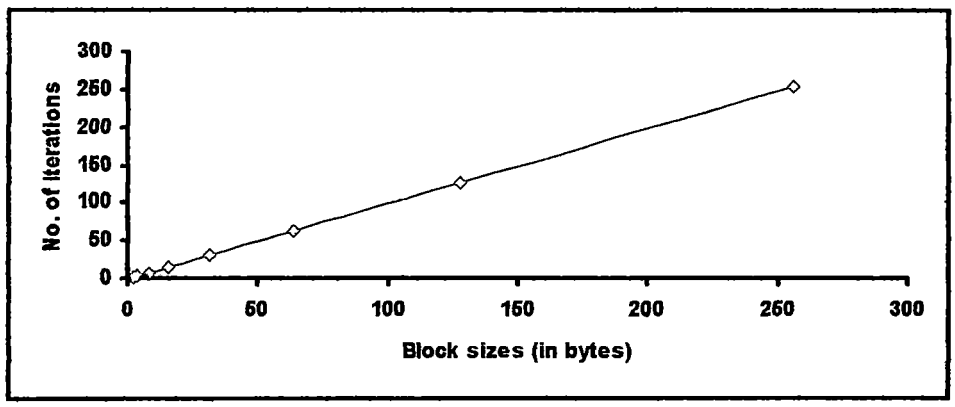

Figure 2.25
Relationship between block size and number of iteration to form a cycle for RCA technique

Table 2.24 shows a steady increment in the number of iterations to complete the cycle if the block size is considered in the form of 2^n. It is seen that for the values of n ranging from 2 to 256, numbers of iterations required are respectively 2, 4, 8, 16, 32, 64, 128, and 256.

Now, the RCA technique may be implemented in an intelligent way by making the blocks to be of different sizes. In that case, different blocks will require different number of iterations to complete the cycle. Naturally the total number of iterations to regenerate the entire stream of bits also will be different.

In this regard, let us consider a simple example. Say, there is a stream of bits of size 128 bits. Using this intelligent approach, the stream is being decomposed into blocks B_1 (8 bits), B_2 (8 bits), B_3 (16 bits), B_4 (32 bits) and B_5 (64 bits). Now, following table 2.24, we get the information that B_1 requires 8 iterations, B_2 requires 8 iterations, B_3 requires 16 iterations, B_4 requires 32 iterations and B_5 requires 64 iterations. Therefore to regenerate the entire stream of bits, the minimum number of iterations required is 64, which is obtained by taking the LCM of 8, 8, 16, 32 and 64. Thus by allowing blocks to be of varying sizes; a far better security can be achieved.

2.5.2 Analysis on factors considered for evaluation purpose

From the results shown in section 2.4, it is clear that for a set of fixed blocks, the execution time (the encryption as well as the decryption) varies almost linearly with the source file size. Now, since any type of source file is being considered simply as a stream

of bits, the encryption (or the decryption) time does not depend on the type of the source file. Hence if the technique is applied for different types of files, all being of around the same size, the encryption (or the decryption) time will be more or less the same for a particular block size or for a particular set of different blocks. Table 2.25 establishes this fact and it is graphically shown in figure 2.26.

In table 2.25, seven files of almost the same size ranging from 70982 bytes to 89786 bytes have been considered. It is observed that using the proposed RCA technique if these files are encrypted, the encryption time is 0.109890 seconds, and the decryption time is also 0.109890 seconds. This means that for these files of almost similar sizes, the encryption/decryption times are also almost similar. Graphically this fact is established in figure 2.26. Bars of blue and red shed represents the encryption time and that of decryption time against file size.

Table 2.25
Result of encryption/decryption time for different types of files of almost same sizes for RCA technique

File name	File size (in bytes)	Encryption time (in seconds)	Decryption time (in seconds)
WAVOTOASF.EXE	83024	0.109890	0.109890
OCCSITE.CPP	89786	0.109890	0.109890
INET.CPP	72980	0.109890	0.109890
BRCC32.EXE	76320	0.109890	0.109890
LVCODEK.SYS	79120	0.109890	0.109890
ODBCINT.DLL	70982	0.109890	0.109890
VSAMI.DLL	81920	0.109890	0.109890

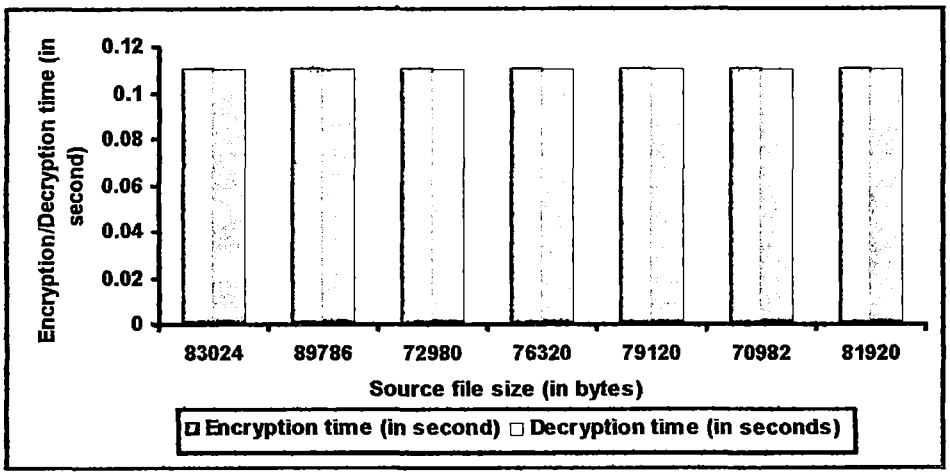

Figure 2.26
Graphical relationship among encryption/decryption times of files of different categories but almost of same sizes for RCA techniques

Results obtained in section 2.4 on Chi square values suggest the fact that it is exactly file-dependent since there exists no fixed relationship between the file size and the Chi square value. Out of all the results obtained, only those for almost the same file sizes have been considered to analyze their Chi square values. Table 2.26 enlists these results and the graphical relationship is shown in figure 2.26.

In table 2.26, seven files have been considered, the sizes ranging from 28685 bytes to 36688 bytes. Two out of these files are with the degree of freedom as 255, two with 224, one with 254 and the remaining are 85 and 219. For these sample files of almost similar sizes, Chi square values range from 102783 to 568921, which means that in spite of the fact the files considered are of almost similar sizes, there Chi square values vary to a large extend. It indicates that the Chi square value is dependent to the content of the file, not to the size of the file.

In figure 2.27, the gray pillars, which are almost of the similar heights, stand for the sizes of sample files, and the corresponding adjacent red pillars, the heights of which vary to a large extend, stand for the corresponding Chi square values.

Table 2.26
Result of Chi square values for different types of files of almost same sizes for RCA technique

File name	File size	Chi square value	Degree of freedom
ENUMVAR.DLL	28685	347732	224
VIEWPREV.CPP	30848	568921	85
TLIB.EXE	32033	300224	255
CLIPSRV.EXE	32768	246256	255
SELFREG.DLL	32256	102783	219
NTIO412.SYS	35392	114958	254
VCARD.DLL	36688	468712	224

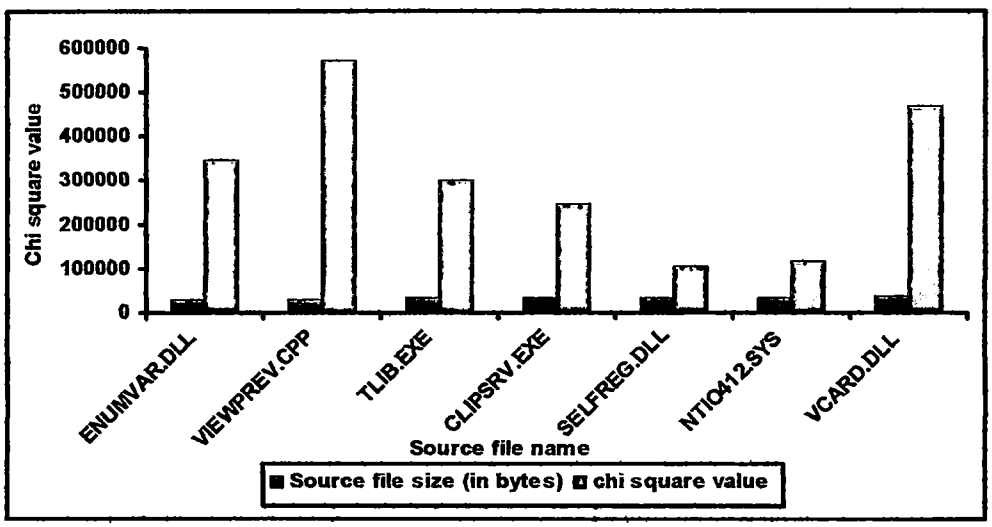

Figure 2.27
Graphical relationship between Chi square values of files of different categories but almost of same sizes

Results of the frequency distribution tests presented in section 2.4.2 suggest the existence of a wide range of distribution of characters in all encrypted files, which is so effective in making it difficult to break the ciphertext using cryptanalysis.

One point to be noted in section 2.4.3 is that the better result is available in terms of the Chi square values for lower block size. But only judging from this angle it cannot be concluded that it is a preferable option to choose smaller blocks, since having a unique size of blocks of only 4 bits makes the implementation of the encryption process so easy.

2.6 Conclusion

Analyzing the proposed RCA technique from different perspectives, it can be pointed out that it is the structure of the scheme that helps in forming a large key space, which in turn helps ensuring the security of a very satisfactory level. If blocks of varying sizes are constructed from a source stream of bits, and for each block arbitrary numbers of iterations are performed during the process of encryption, and if all these information are accommodated in the secret key, then for ensuring the correct decryption, a reasonably long key space is required. One proposal for such a key is presented later in figure 8.2.1 in chapter 8.

Also, section 2.5.1 shows the finiteness in regenerating the source block, which ensures that whatever is the size of a block, after a finite number of iterations it is regenerated, and since this finite number of iterations does not follow any mathematical policy it ensures a better security.

Therefore it can be concluded that the RCA encryption policy itself can produce a satisfactory degree of information security, and, as it is discussed in chapter 9, when it participates in the cascaded approach of encryption, it further enhances the performance a lot.