Comparative Study of Arithmetic and Huffman Compression Techniques for Enhancing Security and Effective Bandwidth Utilization in the Context of ECC for Text

O.Srinivasa Rao
Dept. of CSE,
JNTUK University College of Engineering,
Vizianagaram, A.P.
India-535003

Prof.S.Pallam Setty
CS & SE Dept.,
Andhra University College of Engineering,
Visakapatnam, A.P,
India-530003

ABSTRACT
In this paper, we proposed a model for text encryption using elliptic curve cryptography (ECC) for secure transmission of text and by incorporating the Arithmetic/Huffman data compression technique for effective utilization of channel bandwidth and enhancing the security.

In this model, every character of text message is transformed into the elliptic curve points (X_m,Y_m), these elliptic curve points are converted into cipher text .The resulting size of cipher text becomes *four* times of the original text. For minimizing the channel bandwidth requirements, the encrypted text is compressed using the Arithmetic and Huffman compression technique in the following two ways by considering i)x-y co-ordinates of encrypted text and ii) x-co-ordinates of the encrypted text. The results of the above two cases are compared in terms of overall bandwidth required and saved for Arithmetic and Huffman compression.

Keywords
Elliptic Curve Cryptography (ECC), Text encryption, Huffman compression, Arithmetic compression

1. INTRODUCTION
Over last three decades, the traditional cryptosystem like DES, DLP, AES, DSA and RSA etc. are used for privacy and security. But these conventional methods are not able to support the new generation of digital communication and information access devices, these devices required a crypto-security technology. A method called Elliptic Curve Cryptography is becoming the choice for mobile communication. Elliptic curve cipher use very small key size and computationally is very efficient. N. Koblitz [1] and Victor Miller [2], independently proposed the elliptic curve cryptosystem.

One can use an elliptic curve group that is smaller in size while maintaining the same level of security. The result is smaller key sizes, bandwidth savings, and faster implementations—features that are especially attractive for security applications where computational power and integrated circuit space is limited, such as smart cards, personal digital assistants, and wireless devices. Elliptic curve cryptographic protocols for digital signatures, public-key encryption, and key establishment have been standardized by numerous standards organizations including:

- American National Standards Institute (ANSI X9.62 [3], ANSI X9.63 [4])
- Institute of Electrical and Electronics Engineers (IEEE 1363-2000 [5])
- International Standards Organization (ISO/IEC 15946-3 [6])
- U.S. government's National Institute for Standards and Technology (FIPS 186-2 [7])
- Internet Engineering Task Force (IETF PKIX [7], IETF OAKLEY [8])
- Standards for Efficient Cryptography Group (SECG [9])

The vast majority of the products and standards that use public-key cryptography for encryption and digital signatures use RSA [10]. As we have seen, the bit length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. This burden has ramifications, especially for electronic commerce sites that conduct large numbers of secure transactions. Recently, a competing system that has emerged is elliptic curve cryptosystem (ECC)[4,11].

1.1 Elliptic Curve Cryptography
Elliptic curve cryptography makes use of elliptic curves in which the variables and coefficients are all restricted to elements of a finite field. Two families of elliptic curves are used in cryptographic applications: Prime curves defined over Z_p and binary curves constructed over GF (2^m). Fernandez[12] points out that prime curves are best suited for software applications, as the extended bit –fiddling operations needed by binary curves are not required; ,and that binary curves are best for hardware applications, where it takes remarkably few logic gates to create a powerful and fast cryptosystem. In this paper we used prime curves defined over Z_p for analysis purpose.

1.2Mathematical review
We consider an elliptic curve over prime fields which are of the form:
E: $y^2 = x^3 + ax + b \bmod p$ *where* $a, b \in Fp$ and $4a^3 + 27b^2 \neq 0 \bmod p$

The addition of two points $P(x_1, y_1)$ and $Q(x_2, y_2)$ is calculated by:

$$R(x_3, y_3) = P + Q \text{ where:}$$
$$x_3 = \lambda^2 - x_1 - x_2,$$
$$y_3 = \lambda(x_1 - x_3) - y_1,$$
$$\lambda = (y_2 - y_1)/(x_2 - x_1) \text{ if } P \neq Q$$
$$\lambda = (3x_1^2 + a)/2y_1 \text{ if } P = Q$$

2. DATA COMPRESSION TECHNIQUES

Compression is a technology for reducing the quantity of data used to represent any content without excessively reducing the quality of the picture. It also reduces the number of bits required to store and/or transmit digital media. Compression is a technique that makes storing easier for large amount of data. The performance of data compression algorithms is measured in terms of compression ratio **(CR)** which is defined as

Compression ratio =Size of the output stream/size of the input stream.

We analyzed the adoptability of Huffman data compression techniques for encrypted image data/message in the context of ECC for effective utilization of channel bandwidth for two cases i.e. (i) For different sizes of same image and (ii) For different images of almost same size.

2.1 Arithmetic Coding Technique for Data Compression

Due to its high efficiency and the hardware implementation of Arithmetic coding gains more interest. The basic concept of arithmetic coding can be traced back to Elias in the early 1960s (see [13],pp. 61-621). Practical techniques were first introduced by Rissanen [14] and Pasco [15], and developed further by Rissanen [16]. The reader interested in the broader class of arithmetic codes is referred to [17]; a tutorial is available in [18].

Arithmetic coding is superior in most respects to the better-known Huffman [13] method. In comparison of the Huffman coding algorithm, Arithmetic Coding over comes the constraint that the symbol to be encoded by a whole number of bits. This leads to higher efficiency and a better comparison ratio in general. Indeed Arithmetic coding can be proven to almost reach the best comparison ratio possible, which is bounded by the entropy of the data being encoded. Though during the encoding the algorithm generates one code for the whole input stream, this is done in a fully sequential manner, symbol after symbol.

Huffman Compression Technique

In 1952, Huffman [13] proposed an elegant sequential algorithm which generates optimal prefix codes in O (nlogn) time. The algorithm actually needs only linear time provided that the frequencies of appearances are sorted in advance [20, 21]. Since then there have been extensive researches on analysis, implementation issues and improvements of the Huffman coding theory in a variety of applications [22, 23, 24, 25, 26, 27and 28].

Huffman coding, is a particular method of compressing data through the use of a code table with encodings of variable lengths. A Huffman code is an optimum, or minimum-redundancy, code, which means that messages which occur with greater probability have shorter encodings; in addition, it is prefix free, meaning that no code in the table may be the beginning part of any other code. Huffman describes an algorithm which can be used to generate a binary Huffman code from a collection of messages, or strings, ordered by probability. To generate a code, one starts with a collection of all messages in order of probability. The two least probable messages are removed from the collection and combined into a "composite message," with probability equal to the sum of the messages comprising it. This process is repeated until there is only a single composite message left in the collection, with a probability of 1; that composite message represents the entire Huffman code. This is easily converted to a tree-based approach, in which the initial messages are represented as leaf nodes, each edge represents a digit 0 or 1 in the encoding, and "composite messages" are sub trees created by assigning a common parent to the merged messages.

3. PROPOSED MODEL FOR ENCRYPTION AND DECRYPTION

The proposed model at sender and receiver side for text in the context of ECC for enhancing the security and effective utilization of the channel bandwidth is shown in Figure1.The following two sections describes the proposed model at sender side and at receiver side of text encryption and compression technique for secure transmission of the text by aiming the effective utilization of channel bandwidth.

3.1Encryption and Compression procedure (at sender side)

1. Take plain text X,

2. Each character of X, i.e. assigned as message P_m, can be converted into the point coordinate (X_m, Y_m) on EC

3. Encryption/decryption system require a point on G and an elliptic group $E_p(a, b)$. User A select a private key n_A and generate a public key $P_A = n_A \times G$. To encrypt and send pixel P_m, to B, A choose a random positive integer k and produce the cipher text C_m consisting of the pair of points

$C_m = \{kG, P_m + kP_B\}$—(1), where P_B is the public key of user B.

4. *The* x-coordinates/(x, y) coordinates of *encrypted cipher text values are compressed by using the Arithmetic/Huffman data compression which is then transmitted through in secure channel to the destination.*

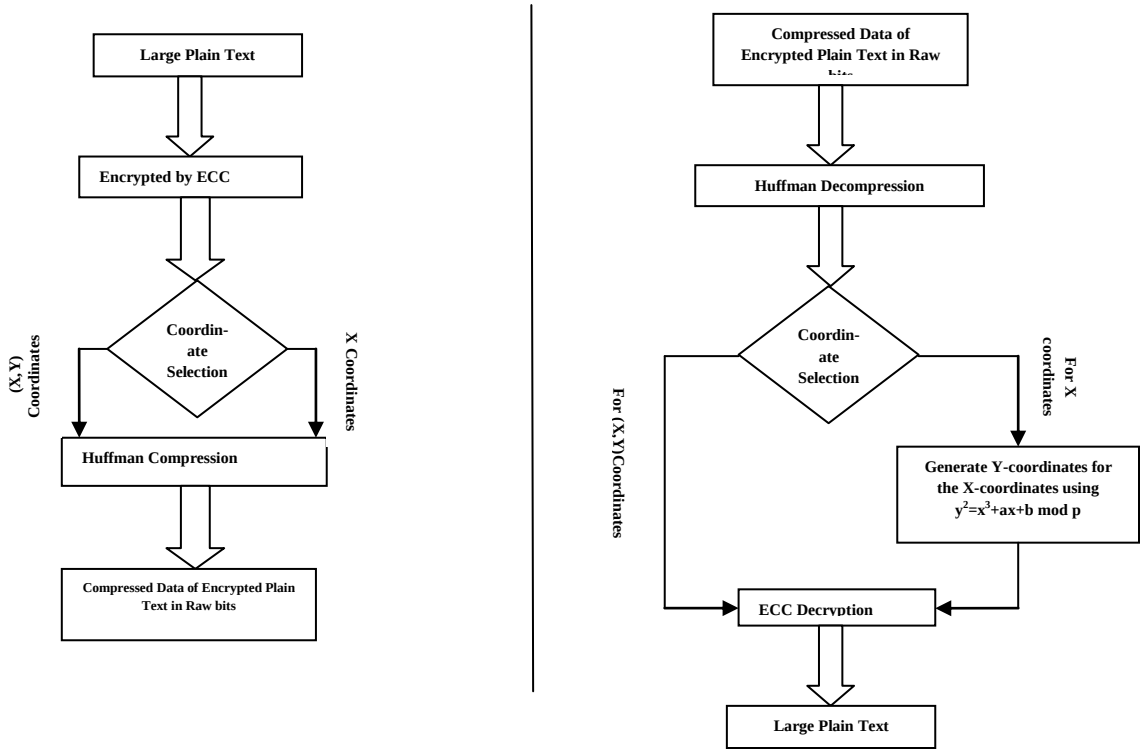
3.1Proposed Decompression and Decryption (at the receiver side)

1. Received *raw data*, i.e. compressed x-coordinates/ (x,y) coordinates of the encrypted text is decompressed using the Arithmetic/Huffman decompression technique

2. To retrieve the cipher text values (*if the raw data contains only x coordinates*), one need to compute y-coordinates also. These values are generated by substituting the x co-ordinate values into the chosen elliptic curve

3. To decrypt the cipher pixel, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m \quad (2)$$



(a) Proposed Encryption & Compression Procedure (At sender side)

b) Proposed Decompression & Decryption Procedure (At Receiver side)

Figure 1 Proposed model at sender side and receiver side for images in the context of ECC

For practical purpose, We have taken an elliptic curve $E_{571}(1,1)$ in the prime field and the alpha numerical characters are mapped [29,30] to the points of the EC. The mapped points are encrypted [31, 32] and computed compression ratio [33] for encrypted points using Huffman, from which we found the overall percentage of the bandwidth required and the percentage of the bandwidth saved.

The following rules are implemented for reducing the bandwidth:

1. The size of encrypted data size is $n*[KG, P_m + KP_B]$ for the n bytes of the message. If, we send all encrypted data as it is to the destination, then the bandwidth required is 4 *n bytes for n byte data message/image, i.e., Four times of the bandwidth required.
2. Instead of sending every point C_m we send only once KG and rest of the $[P_m + KP_B]$ for n times, i.e., for 4n bytes of encrypted data we send only KG+n*[P_m + KP_B] bytes to the destination, which is enough to recover the original Message. The amount of bandwidth saved at this stage is:
   If the n value is very large, then, KG+n(P_m + KP_B)≈ n(P_m + KP_B), hence the reduced bandwidth is give by,

$$\frac{n[P_m + KP_B]}{n[KG(P_m + KP_B)]} = \frac{P_m + KP_B}{KG(P_m + KP_B)}$$

   As we know KG and P_m + KP_B are 1 byte each, so that bandwidth saved to ½ of the originally required, i.e., 50% can be saved at this point.
3. As n is very large the encrypted data of [KG, n *(P_m + KP_B)] will become ≈ n(P_m + KP_B), C_m is compressed using Huffman Compression by considering the following two cases
   (i) Both (x, y) co-ordinates of the encrypted data of [KG + n*(P_m + KP_B)] is compressed using Arithmetic/ Huffman compression and the results are shown in the corresponding tables and graphs [Table 4.1 to Table 4.4 and Figures 4.1 to Figures 4.2]. *In this case, the amount of bandwidth saved is 50% of original encrypted data + reduced size of the compressed data. Hence,*

$$OBWS\% = \frac{0.5 * OEDS + CB\ in\ (x,y)}{OEDS} * 100$$

*OEDS-Original encrypted data Size,

*CB-Compression Bits

The overall percentage of the overall bandwidth required (OBWR) can be calculated by the equation

OBWR%=100-OBWS%

(ii) In this case, only x coordinates of encrypted data of [KG + n*(P_m + KP_B)] is taken for compression, as we know the x-co-ordinate of the ECC, we can get the corresponding y co-ordinate by using the following cubic equation,

$$y^2 \equiv x^3+ax+b \bmod p$$

If we take only x co-ordinate of the original encrypted data, then the amount of bandwidth saved is 75% of original encrypted data + reduced size of the compressed data. Hence, the percentage of the bandwidth saving (OBWS) can be calculated by the equation

$$OBWS\% = \frac{0.75 * OEDS + CB\ in\ (x,y)}{OEDS} * 100$$

The overall percentage of the bandwidth required (OBWR) can be calculated by the equation

OBWR%=100-OBWS%

For this data, we computed the bandwidth required and saved by applying Arithmetic and Huffman compression. The results are shown in tables and graphs.

At the destination the data is uncompressed and original text is recovered by using the equation (2).

4. ARITHMETIC COMPRESSION

The following experiments are conducted by considering only x coordinates and both (*x,y*) Coordinates of the encrypted text for computing compression ratio, overall percentage of bandwidth requirement, and saving for the following three cases.

4.1For (*x, y)* Coordinates of the Encrypted Text

This section presents the compression and compression ratio for three different cases

Table 1: Arithmetic Compression for Case (i)

Sl. No.	Input String		OEDS	(x,y) EDS	Arithmetic	
	String	Size (bits)			CB	CR
1	ABCD	32	128	80	33	0.412
2	ABCDE	40	160	96	42	0.437
3	ABCDEF	48	192	112	53	0.473
4	ABCDEFG	56	224	128	60	0.468
5	ABCDEFGH	64	256	144	61	0.423
6	ABCDEFGHI	72	288	160	61	0.381
7	ABCDEFGHIJ	80	320	176	61	0.346
8	ABCDEFGHIJK	88	352	192	62	0.322
9	ABCDEFGHIJKL	96	384	208	61	0.293
10	ABCDEFGHIJKLM	104	416	224	62	0.276

*EDS-Encrypted data size

Table 2: Arithmetic Compression for Case (ii)

Sl. No.	Input String		OEDS	(x, y) EDS	Arithmetic	
	String	Size			CB	CR
1	APPLE	40	160	96	39	0.406
2	COMPUTER	64	256	144	61	0.423
3	DOCUMENT	64	256	144	61	0.423
4	ELEPHANT	64	256	144	59	0.409
5	GRAPHICS	64	256	144	61	0.423
6	HARDDISK	64	256	144	59	0.409
7	BEAUTIFUL	72	288	160	61	0.381
8	FLOWCHART	72	288	160	61	0.381
9	JNTUCEVZM	72	288	160	61	0.381
10	INFORMATION	88	352	192	59	0.307

Table 3: Arithmetic Compression for Case (iii)

S.No.	Input String		OEDS	(x,y) EDS	Arithmetic	
	String	Size			C	CR
1	AAAABCCDDDD	88	352	192	56	0.29
2	ABCDEEEFFFG	88	352	192	61	0.31
3	ABCDEEEEEEF	88	352	192	61	0.31
4	DDEEFGHHHII	88	352	192	59	0.30
5	KLKLFGHIJJJ	88	352	192	62	0.32
6	AABCCCDHIJK	88	352	192	62	0.32
7	AAABBBCDEFJ	88	352	192	59	0.30
8	AAAABBBBCCC	88	352	192	60	0.31
9	AAAAAAAAAA	96	384	208	36	0.17
10	AAAAAABBBBB	96	384	208	60	0.28

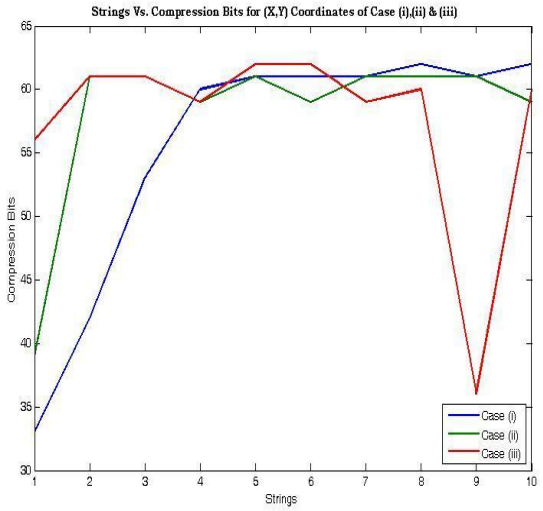


Figure 2: Compression Bits for Case (i),(ii)& (ii)

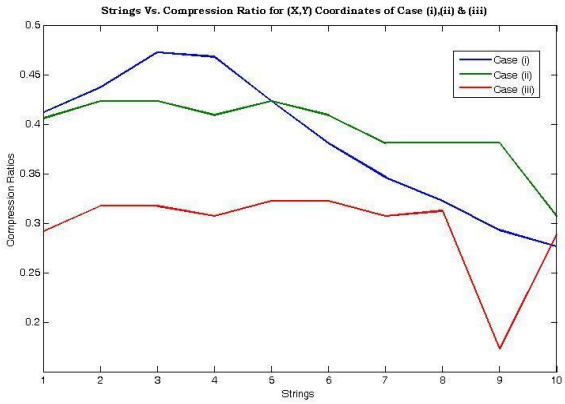


Figure 3: Compression Ratios for Case (i), (ii) & (iii)

One can observe the following variation range in the compression bits and compression ratio from above Tables 1 to 3, Figures 2 and Figure 3

Cases	Compression bits range	Compression Ratio range
Case(i)	33 - 62	0.276 - 0.473
Case(ii)	39 - 61	0.307 – 0.423
Case(iii)	36 - 62	0.173 - 0.322

4.2 Overall BWR% and BWS% in (*x, y*) Coordinates:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering the both (x, y) co-ordinates of the encrypted data

Table 4 Overall BWR% and BWS% in Arithmetic Compression for case (i)

Sl.	Input String	TBS	OBWR%	OBWS%
1	ABCD	95	25.78125	74.21875
2	ABCDE	118	26.25	73.75
3	ABCDEF	139	27.60417	72.39583
4	ABCDEFG	164	26.78571	73.21429
5	ABCDEFGH	195	23.82813	76.17188
6	ABCDEFGHI	227	21.18056	78.81944
7	ABCDEFGHIJ	259	19.0625	80.9375
8	ABCDEFGHIJK	290	17.61364	82.38636
9	ABCDEFGHIJKL	323	15.88542	84.11458
10	ABCDEFGHIJKLM	354	14.90385	85.09615

***TBS-total bits saved=(OEDS-CB)**

Table 5 Overall BWR% and BWS% in Arithmetic Compression for case (ii)

Sl. No.	Input String	TBS	OBWR%	OBWS%
1	APPLE	121	24.375	75.625
2	COMPUTER	195	23.82813	76.171
3	DOCUMENT	195	23.82813	76.171
4	ELEPHANT	197	23.04688	76.953
5	GRAPHICS	195	23.82813	76.171
6	HARDDISK	197	23.04688	76.953
7	BEAUTIFUL	227	21.18056	78.819
8	FLOWCHART	227	21.18056	78.819
9	JNTUCEVZM	227	21.18056	78.819
10	INFORMATION	293	16.76136	83.23864

Table 6 Overall BWR% and BWS% in Arithmetic Compression for case (iii)

Sl. No.	Input String	TBS	OBWR%	OBWS%
1	AAAABCCDDDD	296	15.90909	84.09091
2	ABCDEEEFFFG	291	17.32955	82.67045
3	ABCDEEEEEEF	291	17.32955	82.67045
4	DDEEFGHHHII	293	16.76136	83.23864
5	KLKLFGHIJJJ	290	17.61364	82.38636
6	AABCCCDHIJK	290	17.61364	82.38636
7	AAABBBCDEFJ	293	16.76136	83.23864
8	AAAABBBBCCC	292	17.04545	82.95455
9	AAAAAAAAAAA	348	9.375	90.625
10	AAAAAABBBBBB	324	15.625	84.375

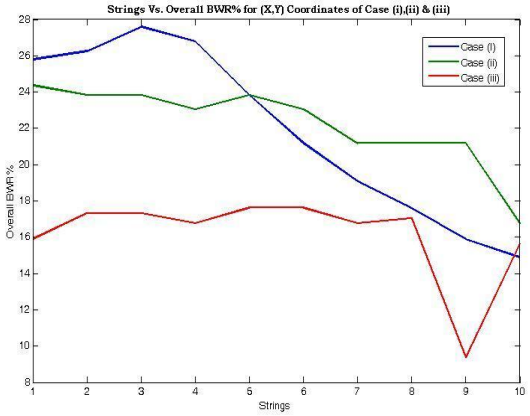


Figure 4: Overall BWR% for Case (i),(ii)& (iii)

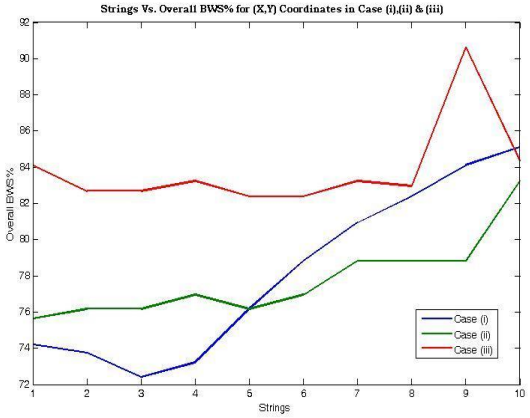


Figure 5 Overall BWS% for Case (i), (ii) & (iii)

One can observe the following variation range in the overall BWR% and BWS% from above Tables 4 to 6, Figures 4 and Figure 5

Cases	OBWR% Range	OBWS% Range
Case(i)	14.90 – 27.60	72.3 – 85.1
Case(ii)	16.76 –24.375	75.625 – 83.23
Case(iii)	9.375 – 17.61	82.38 – 90.625

4.3 For (*x*) Coordinates of the encrypted data:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering only *x* co-ordinates of the encrypted data.

Table 7: Arithmetic Compression for Case (i)

Sl. No.	Input String		OEDS	EDS	Arithmetic	
	String	Size			CB	CR
1	ABCD	32	128	40	12	0.3
2	ABCDE	40	160	48	14	0.29166
3	ABCDEF	48	192	56	19	0.33928
4	ABCDEFG	56	224	64	22	0.34375
5	ABCDEFGH	64	256	72	29	0.40277
6	ABCDEFGHI	72	288	80	33	0.4125
7	ABCDEFGHIJ	80	320	88	38	0.43181
8	ABCDEFGHIJK	88	352	96	42	0.4375
9	ABCDEFGHIJKL	96	384	104	46	0.44230
10	ABCDEFGHIJKLM	104	416	112	53	0.47321

Table 8: Arithmetic Compression for Case (ii)

Sl. No.	Input String		OEDS	EDS	Arithmetic	
	String	Size			CB	CR
1	APPLE	40	160	48	14	0.291666
2	COMPUTER	64	256	72	29	0.402777
3	DOCUMENT	64	256	72	29	0.402777
4	ELEPHANT	64	256	72	27	0.375
5	GRAPHICS	64	256	72	29	0.402777
6	HARDDISK	64	256	72	27	0.375
7	BEAUTIFUL	72	288	80	32	0.4
8	FLOWCHART	72	288	80	33	0.4125
9	JNTUCEVZM	72	288	80	33	0.4125
10	INFORMATION	88	352	96	37	0.385416

Table 9: Arithmetic Compression for Case (iii)

Sl. No.	Input String		OEDS	EDS	Arithmeti	
	String	Size			CB	CR
1	AAAABCCDDDD	88	352	96	25	0.26
2	ABCDEEEFFFG	88	352	96	33	0.34
3	ABCDEEEEEEF	88	352	96	26	0.27
4	DDEEFGHHHII	88	352	96	33	0.34
5	KLKLFGHIJJJ	88	352	96	32	0.33
6	AABCCCDHIJK	88	352	96	35	0.36
7	AAABBBCDEFJ	88	352	96	32	0.33
8	AAAABBBBCCC	88	352	96	20	0.03
9	AAAAAAAAAA	96	384	104	4	0.16
10	AAAAAABBBBB	96	384	104	17	0.22

4.4 Overall BWR% and BWS% in (*x*) Coordinates:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering only x co-ordinates of the encrypted data

Table 10: Overall BWR% and BWS% in Arithmetic Compression case (i)

Sl. No	Input String	TBS(OEDS – CB)	OBWR %	OBWS %
1	ABCD	116	9.375	90.625
2	ABCDE	146	8.75	91.25
3	ABCDEF	173	9.895833	90.10417
4	ABCDEFG	202	9.821429	90.17857
5	ABCDEFGH	227	11.32813	88.67188
6	ABCDEFGHI	255	11.45833	88.54167
7	ABCDEFGHIJ	282	11.875	88.125
8	ABCDEFGHIJK	310	11.93182	88.06818
9	ABCDEFGHIJKL	338	11.97917	88.02083
10	ABCDEFGHIJKL	363	12.74038	87.25962

Table 11: Overall BWR% and BWS% in Arithmetic Compression case (ii)

Sl. No.	Input String	TBS	OBWR%	OBWS%
1	APPLE	146	8.75	91.25
2	COMPUTER	227	11.32813	88.67188
3	DOCUMENT	227	11.32813	88.67188
4	ELEPHANT	229	10.54688	89.45313
5	GRAPHICS	227	11.32813	88.67188
6	HARDDISK	229	10.54688	89.45313
7	BEAUTIFUL	256	11.11111	88.88889
8	FLOWCHART	255	11.45833	88.54167
9	JNTUCEVZM	255	11.45833	88.54167
10	INFORMATION	251	12.84722	87.15278

Table 12: Overall BWR% and BWS% in Arithmetic Compression case (iii)

Sl. No.	Input String	TBS	OBWR%	OBWS%
1	AAAABCCDDDD	327	7.102273	92.89773
2	ABCDEEEFFFG	319	9.375	90.625
3	ABCDEEEEEEF	326	7.386364	92.61364
4	DDEEFGHHHII	319	9.375	90.625
5	KLKLFGHIJJJ	320	9.090909	90.90909
6	AABCCCDHIJK	317	9.943182	90.05682
7	AAABBBCDEFJ	320	9.090909	90.90909
8	AAAABBBBCCC	332	5.681818	94.31818
9	AAAAAAAAAAA	380	1.041667	98.95833
10	AAAAAABBBBBB	367	4.427083	95.5729

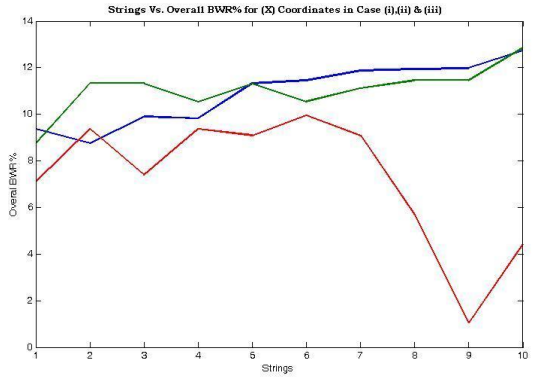


Figure 6: Overall BWR% for Case (i),(ii) & (iii)

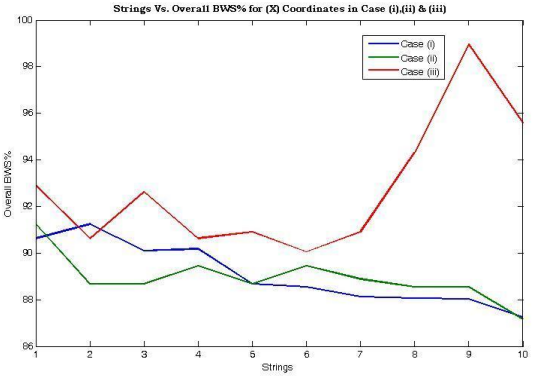


Figure 7 Overall BWS% for Case (i), (ii) & (iii)

One can observe the following variation range in the overall BWR% and BWS% from above Tables 10 to 12, Figures 6 and Figure 7 as follows.

Cases	OBWR% Range	OBWS% Range
Case(i)	8.75- 12.75	87.25 – 91.25
Case(ii)	8.75 – 12.84	87.15 – 91.25
Case(iii)	1.04 – 9.94	90.05 – 98.95

5. HUFFMAN COMPRESSION:

The following experiments are conducted by considering only x coordinates and both (X,Y) Coordinates of the encrypted text for computing compression ratio, the overall percentage of bandwidth required and saved for the following three cases.

5.1For (x, y) Coordinates of the Encrypted Text:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering the both (x,y) co-ordinates of the encrypted data

Table 13: Huffman Compression for Case (i)

Sl. No.	Input String		OEDS	EDS	Huffman	
	String	Size			CB	CR
1	ABCD	32	128	80	34	0.425
2	ABCDE	40	160	96	44	0.458
3	ABCDEF	48	192	112	54	0.482
4	ABCDEFG	56	224	128	64	0.5
5	ABCDEFGH	64	256	144	76	0.527
6	ABCDEFGHI	72	288	160	88	0.55
7	ABCDEFGHIJ	80	320	176	100	0.568
8	ABCDEFGHIJK	88	352	192	112	0.583
9	ABCDEFGHIJKL	96	384	208	124	0.596
10	ABCDEFGHIJKLM	104	416	224	136	0.607

Table 14: Huffman Compression for Case (ii)

Sl. No.	Input String		OEDS	EDS (bits)	Huffman	
	String	Size			CB	CR
1	APPLE	40	160	96	40	0.41
2	COMPUTER	64	256	144	76	0.52
3	DOCUMENT	64	256	144	74	0.51
4	ELEPHANT	64	256	144	72	0.5
5	GRAPHICS	64	256	144	76	0.52
6	HARDDISK	64	256	144	72	0.5
7	BEAUTIFUL	72	288	160	84	0.52
8	FLOWCHART	72	288	160	90	0.56
9	JNTUCEVZM	72	288	160	86	0.53
10	INFORMATIO	88	352	192	100	0.52

Table 15: Huffman Compression for Case (iii)

Sl. No.	Input String		OEDS	EDS (bits)	Huffman	
	String	Size			CB	CR
1	AAAABCCDDDD	88	352	192	76	0.395
2	ABCDEEEFFFG	88	352	192	92	0.479
3	ABCDEEEEEEF	88	352	192	80	0.416
4	DDEEFGHHHII	88	352	192	90	0.467
5	KLKLFGHIJJJ	88	352	192	94	0.489
6	AABCCCDHIJK	88	352	192	98	0.510
7	AAABBBCDEFJ	88	352	192	92	0.479
8	AAAABBBBCCC	88	352	192	70	0.364
9	AAAAAAAAAAA	96	384	208	42	0.201
10	AAAAAABBBBBB	96	384	208	62	0.298

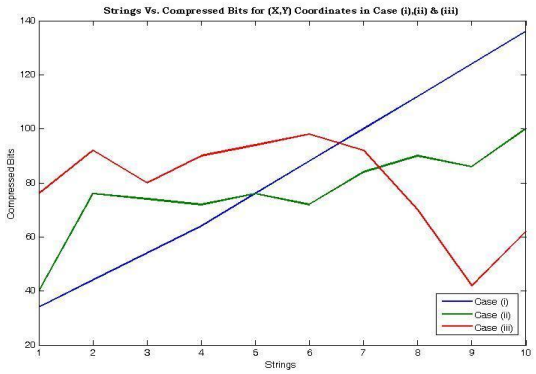


Figure 8: Compressed Bits for Case (i),(ii) & (iii)

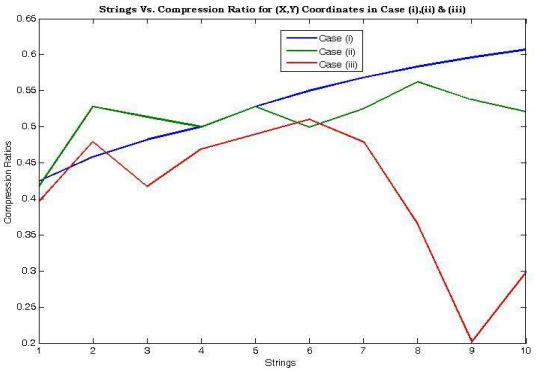


Figure 9: Compressed Ratio for Case (i), (ii) & (iii)

One can observe the following variation range in the compression bits and compression ratio from above the Tables 13 to 15, Figures 8 and Figure 9

Cases	Compression bits	Compression Ratio
Case(i)	34- 136	0.425-0.607
Case(ii)	40-100	0.41 – 0.56
Case(iii)	42-98	0.201-0.51

5.2 Overall BWR% and BWS% in (*x*, *y*) Coordinates:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering the both (x, y) co-ordinates of the encrypted data

Table 16 Overall BWR% and BWS% in Huffman Compression Case (i)

Sl. No	Input String	TBS(OEDS – CB)	OBWR %	OBWS %
1	ABCD	94	26.5625	73.4375
2	ABCDE	116	27.5	72.5
3	ABCDEF	138	28.125	71.875
4	ABCDEFG	160	28.57143	71.42857
5	ABCDEFGH	180	29.6875	70.3125
6	ABCDEFGHI	200	30.55556	69.44444
7	ABCDEFGHIJ	220	31.25	68.75
8	ABCDEFGHIJK	240	31.81818	68.18182
9	ABCDEFGHIJKL	260	32.29167	67.70833
10	ABCDEFGHIJKL	280	32.69231	67.30769

Table 17 Overall BWR% and BWS% in Huffman Compression Case (ii)

Sl. No.	Input String	TBS(OEDS – CB)	OBWR%	OBWS%
1	APPLE	120	25	75
2	COMPUTER	180	29.6875	70.3125
3	DOCUMENT	182	28.90625	71.09375
4	ELEPHANT	184	28.125	71.875
5	GRAPHICS	180	29.6875	70.3125
6	HARDDISK	184	28.125	71.875
7	BEAUTIFUL	204	29.16667	70.83333
8	FLOWCHART	198	31.25	68.75
9	JNTUCEVZM	202	29.86111	70.13889
10	INFORMATION	252	28.40909	71.59091

Table 18: Overall BWR% and BWS% in Huffman Compression Case (iii)

Sl. No	Input String	TBS(OEDS – CB)	OBWR %	OBWS %
1	AAAABCCDDDD	276	21.59091	78.409
2	ABCDEEEFFFG	260	26.13636	73.863
3	ABCDEEEEEEF	272	22.72727	77.27
4	DDEEFGHHHII	262	25.56818	74.43
5	KLKLFGHIJJJ	258	26.70455	73.295
6	AABCCCDHIJK	254	27.84091	72.159
7	AAABBBCDEFJ	260	26.13636	73.863
8	AAAABBBBCCC	282	19.88636	80.113
9	AAAAAAAAAAA	342	10.9375	89.062
10	AAAAAABBBBB	322	16.14583	83.854

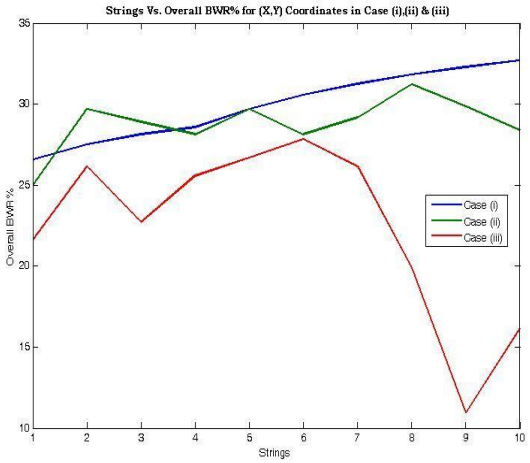


Figure 10: Overall BWR% for Case (i), (ii) & (iii)

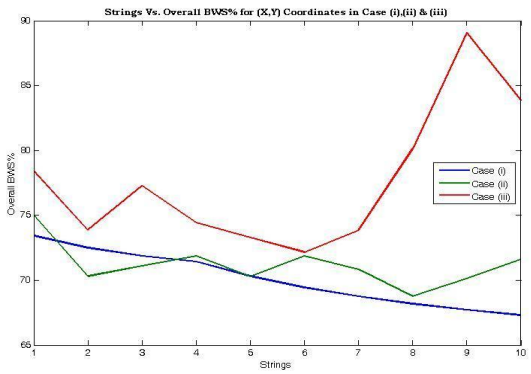


Figure 11: Overall BWS% for Case (i), (ii) & (iii)

One can observe the following variation range the overall BWR% and BWS% from the above Tables 5.4 to 5.6, Figures 5.3 and Figure 5.4.

Cases	Overall BWR%	Overall BWS%
Case(i)	26.56-32.69	67.30- 73.43
Case(ii)	25- 31.25	68.75-75
Case(iii)	10.93-27.84	72.15 – 89.06

5.3 For (X) Coordinates encrypted text data:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering only X co-ordinates of the encrypted data

Table 19: Huffman Compression for Case (i)

Sl. No.	Input String		OEDS	EDS (bits)	Huffman	
	String	Size			CB	CR
1	ABCD	32	128	40	12	0.3
2	ABCDE	40	160	48	16	0.333
3	ABCDEF	48	192	56	20	0.357
4	ABCDEFG	56	224	64	24	0.375
5	ABCDEFGH	64	256	72	29	0.402
6	ABCDEFGHI	72	288	80	34	0.425
7	ABCDEFGHIJ	80	320	88	39	0.443
8	ABCDEFGHIJK	88	352	96	44	0.458
9	ABCDEFGHIJKL	96	384	104	49	0.471
10	ABCDEFGHIJKLM	104	416	112	54	0.482

Table 20: Huffman Compression for Case (ii)

Sl. No.	Input String		OEDS	EDS (bits)	Huffman	
	String	Size			CB	CR
1	APPLE	40	160	48	14	0.291666
2	COMPUTER	64	256	72	29	0.402777
3	DOCUMENT	64	256	72	29	0.402777
4	ELEPHANT	64	256	72	27	0.375
5	GRAPHICS	64	256	72	29	0.402777
6	HARDDISK	64	256	72	27	0.375
7	BEAUTIFUL	72	288	80	32	0.4
8	FLOWCHART	72	288	80	34	0.425
9	JNTUCEVZM	72	288	80	34	0.425
10	INFORMATION	88	352	96	38	0.39583

Table 21: Huffman Compression for Case (iii)

Sl. No.	Input String		OEDS	EDS (bits)	Huffman	
	String	Size			CB	CR
1	AAAABCCDDDD	88	352	96	26	0.27
2	ABCDEEEFFFG	88	352	96	34	0.35
3	ABCDEEEEEEF	88	352	96	28	0.29
4	DDEEFGHHHII	88	352	96	33	0.34
5	KLKLFGHIJJJ	88	352	96	35	0.36
6	AABCCCDHIJK	88	352	96	37	0.38
7	AAABBBCDEFJ	88	352	96	34	0.35
8	AAAABBBBCCC	88	352	96	24	0.12
9	AAAAAAAAAAA	96	384	104	13	0.19
10	AAAAAABBBBB	96	384	104	20	0.25

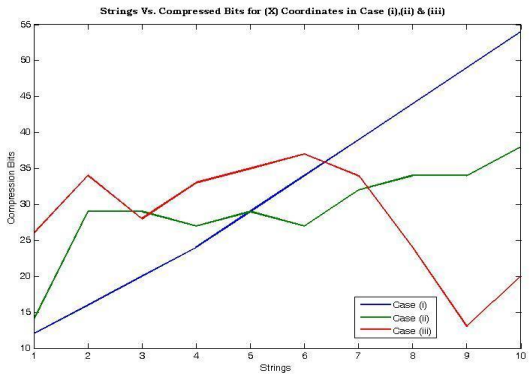


Figure 12: Compressed Bits for Case (i), (ii) & (iii)

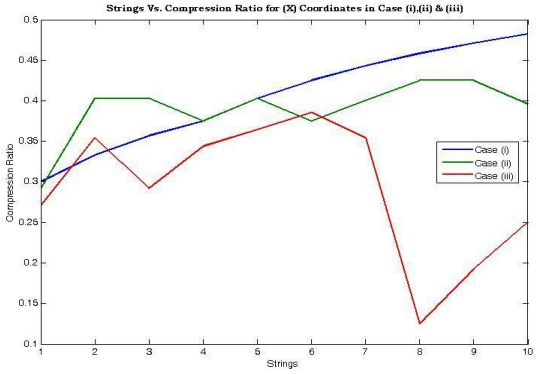


Figure 13: Compressed Ratio for Case (i), (ii) & (iii)

One can observe the following variation range in the compression bits and compression ratio from the above Tables 19 to 21, Figures 12 and Figure 13

Cases	Compression bits	Compression Ratio
Case(i)	12-54	0.3-0.48
Case(ii)	14-38	0.29 – 0.425
Case(iii)	13-37	0.125-0.385

5.4 Overall BWR% and BWS% in *x* Coordinates:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering only x co-ordinates of the encrypted data

Table 22: Overall BWR% and BWS% in Huffman Compression Case (i)

Sl. No.	Input String	TBS	OBWR%	OBWS%
1	ABCD	116	9.375	90.625
2	ABCDE	144	10	90
3	ABCDEF	172	10.41667	89.583
4	ABCDEFG	200	10.71429	89.285
5	ABCDEFGH	227	11.32813	88.671
6	ABCDEFGHI	254	11.80556	88.194
7	ABCDEFGHIJ	281	12.1875	87.812
8	ABCDEFGHIJK	308	12.5	87.5
9	ABCDEFGHIJKL	335	12.76042	87.239
10	ABCDEFGHIJKLM	362	13.94231	86.057

Table 23: Overall BWR% and BWS% in Huffman Compression Case (ii)

Sl. No.	Input String	TBS	OBWR%	OBWS%
1	APPLE	146	8.75	91.25
2	COMPUTER	227	11.32813	88.67188
3	DOCUMENT	227	11.32813	88.67188
4	ELEPHANT	229	10.54688	89.45313
5	GRAPHICS	227	11.32813	88.67188
6	HARDDISK	229	10.54688	89.45313
7	BEAUTIFUL	256	11.11111	88.88889
8	FLOWCHART	254	11.80556	88.19444
9	JNTUCEVZM	254	15.97222	84.02778
10	INFORMATION	314	10.7954	89.2045

Table 24: Overall BWR% and BWS% in Huffman Compression Case (iii)

Sl. No	Input String	TBS	OBWR %	OBWS %
1	AAAABCCDDDD	326	7.386364	92.61364
2	ABCDEEEFFFG	318	9.659091	90.34091
3	ABCDEEEEEEF	324	7.954545	92.04545
4	DDEEFGHHHII	319	9.375	90.625
5	KLKLFGHIJJJ	317	9.943182	90.05682
6	AABCCCDHIJK	315	10.51136	89.48864
7	AAABBBCDEFJ	318	9.659091	90.34091
8	AAAABBBBCCC	328	6.818182	93.18182
9	AAAAAAAAAAA	371	3.385417	96.61458
10	AAAAAABBBBB	364	5.208333	94.79167

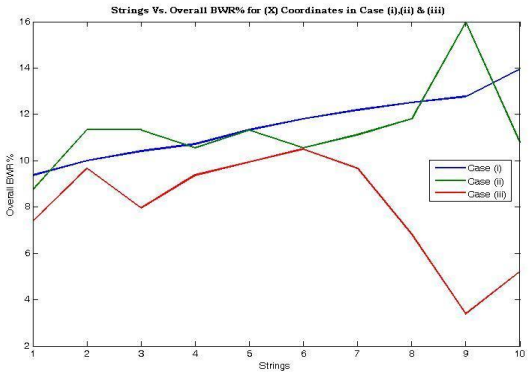


Figure 14 : Overall BWR% for Case (i), (ii) & (iii)

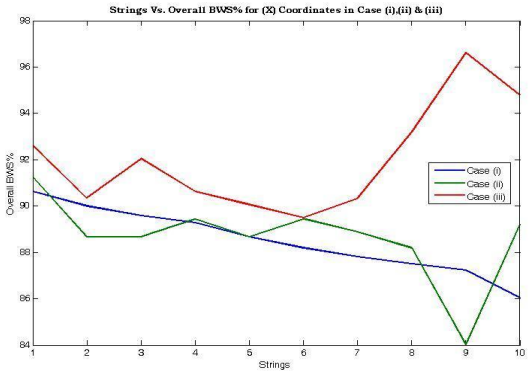


Figure 15: Overall BWS% for Case (i), (ii) & (iii)

One can observe the following variation range in the overall BWR% and BWS% from the above Tables 22 to 24 and Figures 14 and Figure 15 as follows.

Cases	OBWR% Range	OBWS% Range
Case(i)	9.375-13.94	86.05-90.625
Case(ii)	8.75 – 15.97	84.02 – 91.25
Case(iii)	3.38 – 10.51	89.48 – 96.61

6. ANALYSIS OF ARITHMETIC AND HUFFMAN COMPRESSION FOR OVERALL BWR% AND BWS%

This section compares the performance of Arithmetic and Huffman compression Techniques

6.1 For (x, y) Coordinates of the encrypted data:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering the both (x,y) co-ordinates of the encrypted data

Table 25: Overall BWR% and BWS% for case(i)

Sl. No	Input String	BWR%		BWS%	
		Arith	Huff	Arith	Huff
1	ABCD	25.78	26.5	74.21	73.4
2	ABCDE	26.25	27.5	73.75	72.5
3	ABCDEF	27.60	28.1	72.39	71.8
4	ABCDEF	26.78	28.5	73.21	71.4
5	ABCDEF	23.82	29.6	76.17	70.3
6	ABCDEF	21.18	30.5	78.81	69.4
7	ABCDEF	19.06	31.2	80.93	68.7
8	ABCDEF	17.61	31.8	82.38	68.1
9	ABCDEF	15.88	32.2	84.11	67.7
1	ABCDEF	14.90	32.6	85.09	67.3

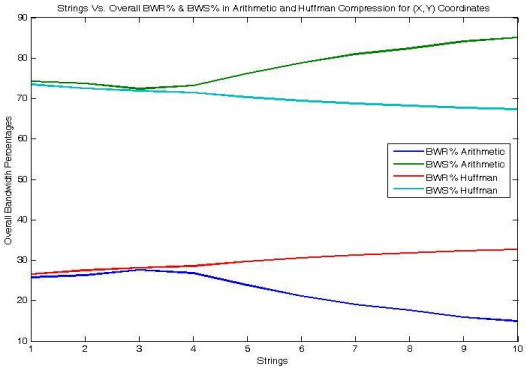


Figure 16: Overall BWR% & BWS%

From the above Table 25and Figure 16, one can observe that, the overall percentage variation range in the bandwidth requirement and saving as follows:

S.No.	Compression Type	BWR% range	BWS% range
1	Arithmetic	14.92 – 27.6	72.39 – 85.09
2	Huffman	26.56 – 32.69	67.32 -73.43

Table 26: Overall BWR% and BWS% for case (ii)

Sl. No.	Input String	OBWR%		OBWS%	
		Arithmetic	Huffman	Arithmetic	Huffman
1	APPLE	24.375	25	75.625	75
2	COMPUTER	23.82813	29.6875	76.1718	70.312
3	DOCUMENT	23.82813	28.90625	76.1718	71.093
4	ELEPHANT	23.04688	28.125	76.9531	71.87
5	GRAPHICS	23.82813	29.6875	76.1718	70.312
6	HARDDISK	23.04688	28.125	76.9531	71.875
7	BEAUTIFUL	21.18056	29.16667	78.8194	70.833
8	FLOWCHART	21.18056	31.25	78.8194	68.75
9	JNTUCEVZM	21.18056	29.86111	78.8194	70.138
10	INFORMATION	16.76136	28.40909	83.2386	71.590

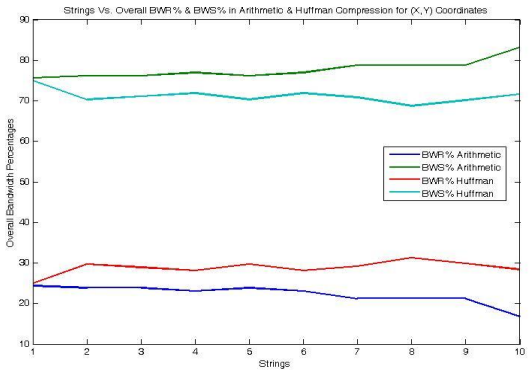


Figure 17: Overall BWR% & BWS%

From the above Table 26 and Figure 17, one can observe that, the overall percentage variation range in the bandwidth requirement and saving as follows:

S.No.	Compression Type	BWR% range	BWS% range
1	Arithmetic	16.76 – 24.375	75.625 – 83.23
2	Huffman	25 – 31.25	68.75 -75

Table 27: Overall BWR% and BWS% for case (iii)

Sl. No	Input String	OBWR%	OBWS %	OBWR%	OBWS %
		Arithmetic	Huffman	Arithmetic	Huffman
1	AAAAAAAAAAA	15.90909	21.5909	84.09091	78.4090
2	AAAAAABBBBBB	17.32955	26.1363	82.67045	73.8636
3	AAAABBBCCCCC	17.32955	22.7272	82.67045	77.2727
4	AAABBBCCCDDD	16.76136	25.5681	83.23864	74.4318
5	AABBCCDDEEFF	17.61364	26.7045	82.38636	73.2954
6	AABBCCDDDEEE	17.61364	27.8409	82.38636	72.1590
7	AAAABCCDDDD	16.76136	26.1363	83.23864	73.8636
8	ABCDEEEFFFG	17.04545	19.8863	82.95455	80.1136
9	ABCDEEEEEEF	9.375	10.9375	90.625	89.0625
10	DDEEFGHHHII	15.625	16.1458	84.375	83.8541

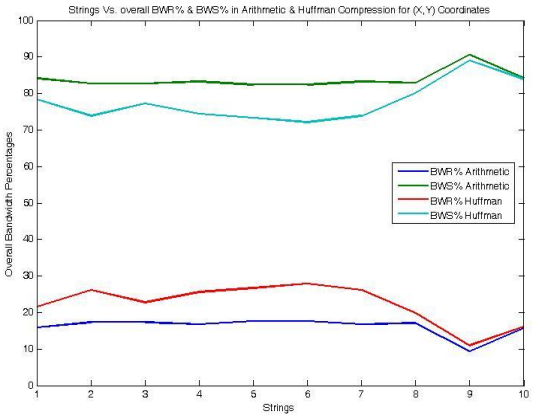


Figure 18: Overall BWR% & BWS%

From the above Table 27 and Figure 18, one can observe that, the overall percentage variation range in the bandwidth requirement and saving as follows:

S.No.	Compression Type	BWR% range	BWS% range
	Arithmetic	9.375 – 17.61	82.38 – 90.625
2	Huffman	10.93 – 27.84	72.15 -89.06

6.2 For (*x*) Coordinates of the encrypted data:

This section gives the overall percentage of bandwidth requirement and saved for above three cases by considering only X co-ordinates of the encrypted data

Table 28: Overall BWR% and BWS% for case(i)

Sl. No.	Input String	BWR%		BWS%	
		Arithmetic	Huffman	Arithmetic	Huffman
1	ABCD	9.375	9.375	90.625	90.625
2	ABCDE	8.75	10	91.25	90
3	ABCDEF	9.895833	10.41667	90.10417	89.58333
4	ABCDEFG	9.821429	10.71429	90.17857	89.28571
5	ABCDEFGH	11.32813	11.32813	88.67188	88.67188
6	ABCDEFGHI	11.45833	11.80556	88.54167	88.19444
7	ABCDEFGHIJ	11.875	12.1875	88.125	87.8125
8	ABCDEFGHIJK	11.93182	12.5	88.06818	87.5
9	ABCDEFGHIJKL	11.97917	12.76042	88.02083	87.23958
10	ABCDEFGHIJKLM	12.74038	13.94231	87.25962	86.05769

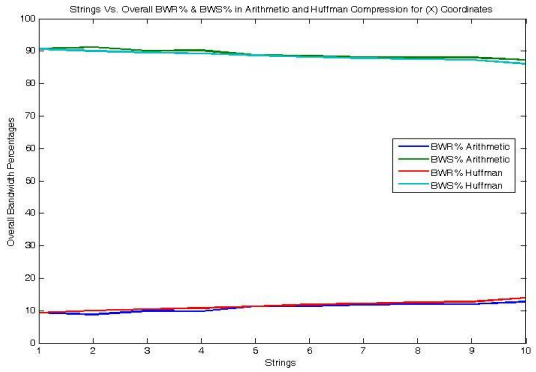


Figure 19: Overall BWR% and BWS%

From the above Table 28 and Figure 19, one can observe that, the overall percentage variation range in the bandwidth requirement and saving as follows:

S.No.	Compression Type	BWR% range	BWS% range
1	Arithmetic	8.75 – 12.74	87.25 – 91.25
2	Huffman	9.375 – 13.94	86.05 -90.625

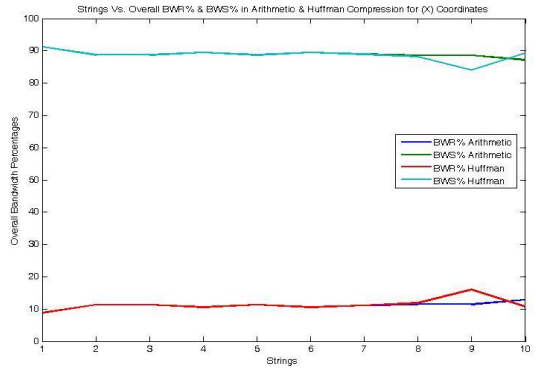


Figure 20: Overall BWR% and BWS%

Table 29 Overall BWR% and BWS% for case(i)

Sl. No.	Input String	OBWR%		OBWR%	
		Arithmetic	Huffman	Arithmetic	Huffman
1	APPLE	8.75	8.75	91.25	91.25
2	COMPUTER	11.32813	11.32813	88.67188	88.67188
3	DOCUMENT	11.32813	11.32813	88.67188	88.67188
4	ELEPHANT	10.54688	10.54688	89.45313	89.45313
5	GRAPHICS	11.32813	11.32813	88.67188	88.67188
6	HARDDISK	10.54688	10.54688	89.45313	89.45313
7	BEAUTIFUL	11.11111	11.11111	88.88889	88.88889
8	FLOWCHART	11.45833	11.80556	88.54167	88.19444
9	JNTUCEVZM	11.45833	15.97222	88.54167	84.02778
10	INFORMATION	12.84722	10.7954	87.15278	89.2045

From the above Table 29 and Figure 20, one can observe that, the overall percentage variation range in the bandwidth requirement and saving as follows:

S.No.	Compression Type	BWR% range	BWS% range
1	Arithmetic	8.75 – 12.84	87.15 – 91.25
2	Huffman	8.75 – 15.97	84.02 -91.25

Table 30 Overall BWR% and BWS% for case(iii):

Sl. No.	Input String	BWR%		BWS%	
		Arithmetic	Huffman	Arithmetic	Huffman
	AAAAAAAAAAA	7.102273	7.386364	92.89773	92.61364
2	AAAAAABBBBBB	9.375	9.659091	90.625	90.34091
3	AAAABBBBCCCC	7.386364	7.954545	92.61364	92.04545
4	AAABBBCCCDDD	9.375	9.375	90.625	90.625
5	AABBCCDDEEFF	9.090909	9.943182	90.90909	90.05682
6	AABBCCDDDEEE	9.943182	10.51136	90.05682	89.48864
7	AAAABCCDDDD	9.090909	9.659091	90.90909	90.34091
8	ABCDEEEFFFG	5.681818	6.818182	94.31818	93.18182
9	ABCDEEEEEEF	1.041667	3.385417	98.95833	96.61458
10	DDEEFGHHHII	4.427083	5.208333	95.57292	94.79167

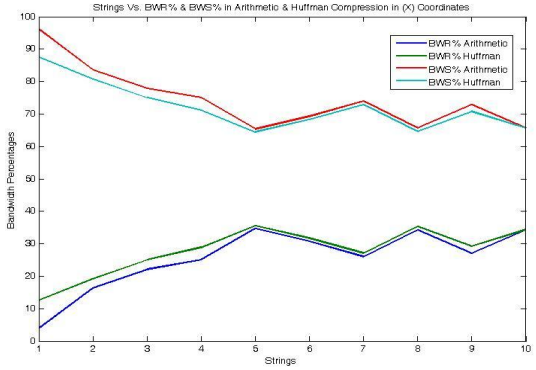


Figure 21: Overall BWR% & BWS%

From the above Table 30 and Figure 21, one can observe that, the overall percentage variation range in the bandwidth requirement and saving as follows:

S.No.	Compression Type	BWR% range	BWS% range
1	Arithmetic	1.04 – 9.94	90.05 – 98.95
2	Huffman	1.38 – 10.51	89.48-96.31

7. Analysis of Overall BWR% for Arithmetic and Huffman Compressions in (X,Y) and (X) Coordinates of Encrypted Data

Table 31: Overall BWR% for (X,Y) Vs. X co-ordinates in Huffman compression & Arithmetic compression for Case (i)

Sl. No.	Input String	Huffman		Arithmetic	
		BWR% (X,Y)	BWR% (X)	BWR% (X,Y)	BWR% (X)
	ABCD	26.5625	9.375	25.78125	9.375
2	ABCDE	27.5	10	26.25	8.75
3	ABCDEF	28.125	10.41667	27.60417	9.895833
4	ABCDEFG	28.57143	10.71429	26.78571	9.821429
5	ABCDEFGH	29.6875	11.32813	23.82813	11.32813
6	ABCDEFGHI	30.55556	11.80556	21.18056	11.45833
7	ABCDEFGHIJ	31.25	12.1875	19.0625	11.875
8	ABCDEFGHIJK	31.81818	12.5	17.61364	11.93182
9	ABCDEFGHIJKL	32.29167	12.76042	15.88542	11.97917
10	ABCDEFGHIJKLM	32.69231	13.94231	14.90385	12.74038

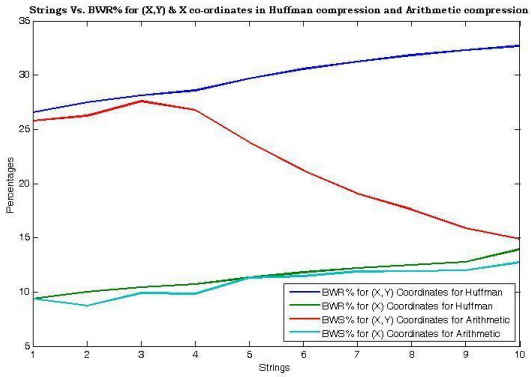


Figure 22: Strings Vs. BWR% for (X,Y) & (X) co-ordinates in Huffman compression And Arithmetic compression

From the above Table 31 and Figure 22 one can observe the overall percentage of bandwidth requirement variation range in (x,y) and x co-ordinates as follows.

S.No.	Compression	BWR% range	
		(x,y)	x
1	Arithmetic	14.92-27.6	8.75 – 12.74
2	Huffman	26.56-32.69	9.37- 13.94

Table 32: Overall BWR% for (X,Y) Vs. X co-ordinates in Huffman compression & Arithmetic compression for Case (ii)

Sl. No.	Input String	Huffman		Arithmetic	
		BWR% (X,Y)	BWR% (X)	BWR% (X,Y)	BWR % (X)
1	APPLE	25	8.75	24.375	8.75
2	COMPUTER	29.687	11.328	23.828	11.328
3	DOCUMENT	28.906	11.328	23.828	11.328
4	ELEPHANT	28.125	10.546	23.046	10.546

5	GRAPHICS	29.687	11.328	23.828	11.328
6	HARDDISK	28.125	10.546	23.046	10.546
7	BEAUTIFUL	29.166	11.111	21.180	11.111
8	FLOWCHART	31.25	11.805	21.180	11.458
9	JNTUCEVZM	29.861	15.972	21.180	11.458
10	INFORMATION	28.409	10.795	16.761	12.847

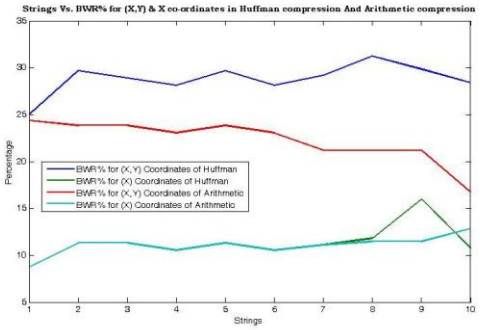


Figure 23: Strings Vs. BWR% for (X,Y) & (X) co-ordinates in Huffman compression And Arithmetic compression of overall

From the above Table 32 and Figure 23 one can observe the overall percentage of bandwidth requirement variation range in (x,y) and x co-ordinates as follows.

S.No.	Compression	BWR% range	
		(x,y)	x
1	Arithmetic	16.76– 24.37	8.75 – 12.84
2	Huffman	25 – 31.25	8.75 – 15.97

Table 33: Overall BWR% for (X,Y) Vs. X co-ordinates in Huffman compression And Arithmetic compression for Case (iii)

Sl. No.	Input String	Huffman		Arithmetic	
		BWR% (X,Y)	BWR% (X)	BWR% (X,Y)	BWR% (X)
	AAAABCCDDDD	21.59091	7.386364	15.90909	7.102273
2	ABCDEEEFFFG	26.13636	9.659091	17.32955	9.375
3	ABCDEEEEEEF	22.72727	7.954545	17.32955	7.386364
4	DDEEFGHHHII	25.56818	9.375	16.76136	9.375
5	KLKLFGHIJJJ	26.70455	9.943182	17.61364	9.090909
6	AABCCCDHIJK	27.84091	10.51136	17.61364	9.943182
7	AAABBBCDEFJ	26.13636	9.659091	16.76136	9.090909
8	AAAABBBBCCC	19.88636	6.818182	17.04545	5.681818
9	AAAAAAAAAAA	10.9375	3.385417	9.375	1.041667
10	AAAAAABBBBBB	16.14583	5.208333	15.625	4.427083

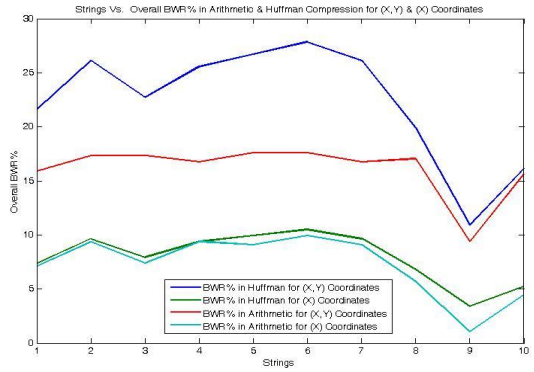


Figure 24: Strings Vs. BWR% for (X,Y) & (X) co-ordinates in Huffman compression And Arithmetic compression

From the above Table 33 and Figure 24 one can observe the overall percentage of bandwidth requirement variation range in (x,y) and x co-ordinates as follows.

S.No.	Compression	BWR% range	
		(x,y)	x
1	Arithmetic	9.375 -17.61	1.04 – 9.94
2	Huffman	10.93 – 27.84	3.38-10.5

8. ANALYSIS OF OVERALL BWS% FOR ARITHMETIC AND HUFFMAN COMPRESSIONS IN (X,Y) AND (X) COORDINATES OF ENCRYPTED DATA

Table 34: Overall BWS% for (X,Y) Vs. X co-ordinates in Huffman compression & Arithmetic compression for Case (i)

Sl. No.	Input String	Huffman		Arithmetic	
		BWS% (X,Y)	BWS% (X)	BWS% (X,Y)	BWS% (X)
1	ABCD	73.4375	90.625	74.21875	90.625
2	ABCDE	72.5	90	73.75	91.25
3	ABCDEF	71.875	89.58333	72.39583	90.10417
4	ABCDEFG	71.42857	89.28571	73.21429	90.17857
5	ABCDEFGH	70.3125	88.67188	76.17188	88.67188
6	ABCDEFGHI	69.44444	88.19444	78.81944	88.54167
7	ABCDEFGHIJ	68.75	87.8125	80.9375	88.125
8	ABCDEFGHIJK	68.18182	87.5	82.38636	88.06818
9	ABCDEFGHIJKL	67.70833	87.23958	84.11458	88.02083
10	ABCDEFGHIJKLM	67.30769	86.05769	85.09615	87.25962

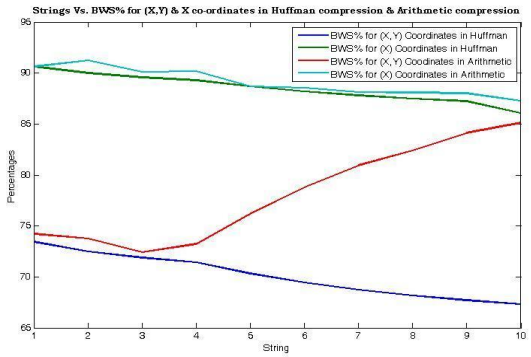


Figure 25: Strings Vs. BWS% for (X,Y) & X co-ordinates in Huffman compression And Arithmetic compression

From the above Table 34 and Figure 25 one can observe that, the overall percentage of bandwidth saving variation range for (x,y) and x co-ordinates as follows.

S.No.	Compression	BWS% range	
		(x,y)	x
1	Arithmetic	72.39 - 85.09	87.25 – 91.25
2	Huffman	67.3 – 73.43	86.05 – 90.625

Table 36: Overall BWS% for (X,Y) Vs. X co-ordinates in Huffman compression And Arithmetic compression for Case (ii)

Sl. No.	Input String	Huffman		Arithmetic	
		BWS% (X,Y)	BWS% (X)	BWS% (X,Y)	BWS% (X)
	APPLE	75	91.25	75.625	91.25
2	COMPUTER	70.3125	88.67188	76.17188	88.67188
3	DOCUMENT	71.09375	88.67188	76.17188	88.67188
4	ELEPHANT	71.875	89.45313	76.95313	89.45313
5	GRAPHICS	70.3125	88.67188	76.17188	88.67188
6	HARDDISK	71.875	89.45313	76.95313	89.45313
7	BEAUTIFUL	70.83333	88.88889	78.81944	88.88889
8	FLOWCHART	68.75	88.19444	78.81944	88.54167
9	JNTUCEVZM	70.13889	84.02778	78.81944	88.54167
10	INFORMATION	71.59091	89.2045	83.23864	87.15278

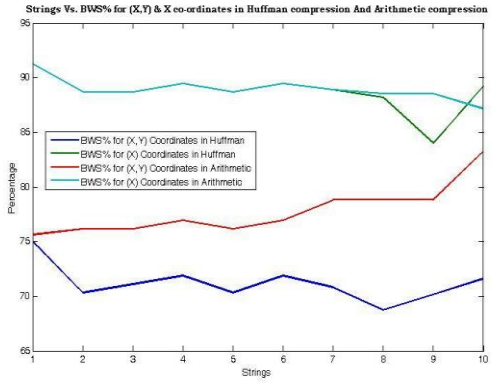


Figure 26: Strings Vs. BWS% for (X,Y) & (X) co-ordinates in Huffman compression And Arithmetic compression

From the above Table 36 and Figure 26 one can observe that, the overall percentage of bandwidth saving variation range for (x,y) and x co-ordinates as follows.

S.No.	Compression	BWS% range	
		(x,y)	x
1	Arithmetic	75.625- 83.23	87.15 – 91.25
2	Huffman	68.75 – 75	84.04 – 91.25

Table 36: Overall BWS% for (X,Y) Vs. X co-ordinates in Huffman compression And Arithmetic compression Case (iii)

Sl. No.	Input String	Huffman		Arithmetic	
		BWS% (X,Y)	BWS% (X)	BWS% (X,Y)	BWS% (X)
	AAAABCCDDDD	78.40909	92.61364	84.09091	92.89773
2	ABCDEEEFFFG	73.86364	90.34091	82.67045	90.625
3	ABCDEEEEEEF	77.27273	92.04545	82.67045	92.61364
4	DDEEFGHHHII	74.43182	90.625	83.23864	90.625
5	KLKLFGHIJJJ	73.29545	90.05682	82.38636	90.90909
6	AABCCCDHIJK	72.15909	89.48864	82.38636	90.05682
7	AAABBBCDEFJ	73.86364	90.34091	83.23864	90.90909
8	AAAABBBBCCC	80.11364	93.18182	82.95455	94.31818
9	AAAAAAAAAAAA	89.0625	96.61458	90.625	98.95833
10	AAAAAABBBBBB	83.85417	94.79167	84.375	95.57292

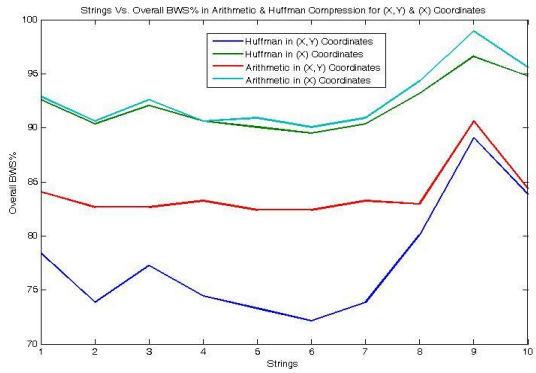


Figure 27: Strings Vs. BWS% for (X,Y) Vs. (X) co-ordinates in Huffman compression And Arithmetic compression

From the above Table 36 and Figure 27 one can observe that, the overall percentage of bandwidth saving variation range for (x,y) and x co-ordinates as follows.

S.No.	Compression	BWS% range	
		(x,y)	x
1	Arithmetic	72.15- 89.06	89.48 – 96.61
2	Huffman	82.67– 91.625	90.05-98.95

9. CONCLUSION

We have conducted the experiments for the following cases, by considering only *x* co-ordinate and both *(x,y)* co-ordinates of the different encrypted text for transmission in Arithmetic as well as in Huffman for small text and as the Arithmetic compression is not suitable for large text, we consider only Huffman compression :

For small text, irrespective of the case, when we consider both (x,y) coordinates for transmission, the overall Percentage of bandwidth requirement varies from 10.93% to 32.69% and the percentage of Bandwidth Saving (BWS) varies from 67.3% to 89.06% in Huffman whereas the overall percentage of bandwidth requirement varies from 9.375% to 27.6% and the percentage of Bandwidth Saving (BWS) varies from 72.39% to 90.625% in Arithmetic .

When we consider only *x* co-ordinate for transmission, the overall percentage of bandwidth requirement varies from 3.38% to 15.97% and the percentage of Bandwidth Saving (BWS) varies from *84.02% to 96.61%* in Huffman, whereas the overall percentage of bandwidth requirement varies from 1.04% to 12.84% and the percentage of Bandwidth Saving (BWS) varies from *87.15% to 98.95%,* in Arithmetic.

Hence, from the above experimental observations, we conclude that, *irrespective of the data, Arithmetic Compression is more suitable for small text when compared with Huffman compression and for large text Huffman compression is suitable.*

10. REFERENCES

[1] Neal Koblitz, "Elliptic Curve Cryptosystem, Journal of mathematics computation Vol.48, No.177pp.203-209,Jan-1987.

[2] V. Miller, "Uses of elliptic curves in cryptography", Advances in Cryptology–Crypto '85,Lecture Notes in Computer Science, 218

[3] Certicom Corp., " An Introduction to Information Security", Number 1, March 1997.

[4] ANSI X9.63, Public Key Cryptography for the Financial Services Industry: Elliptic CurveKey Agreement and Key Transport Protocols, ballot version, May 2001.

[5] Internet Engineering Task Force, The OAKLEY Key Determination Protocol, IETF RFC 2412, November 1998.

[6] ISO/IEC 15946-3, Information Technology–Security Techniques–Cryptographic TechniquesBased on Elliptic Curves, Part 3, Final Draft International Standard (FDIS), February 2001

[7] National Institute of Standards and Technology, Digital Signature Standard, FIPS Publication186-2, 2000.

[8] M. Jacobson, N. Koblitz, J. Silverman, A. Stein and E. Teske, "Analysis of the xedni calculus attack", Designs, Codes and Cryptography, 20 (2000), 41-64. (1986), Springer-Verlag, 417-426.

[9] Standards for Efficient Cryptography Group, SEC 1: Elliptic Curve Cryptography, version1.0, 2000. Available at http://www.secg.org

[10] R.L. Rivest, A. Shamir, and L.M. Adleman, Method for Obtaining Digital Signatures and Public-key Cryptosystems", Communications of the ACM,Volume 21, pages 120-126, February 1978.

[11] S. Arita, "Weil descent of elliptic curves over finite fields of characteristic three", Advances in Cryptology–Asiacrypt 2000, LectureNotes in Computer Science, 1976 (2000),Springer-Verlag, 248-259

[12] Fernandes, A. "Elliptic Curve Cryptography", Dr.Dobb's journal, December 1999

[13] D. A. Huffman, "A method for the construction of minimum redundancy codes", Proc. IRE, Vol. 40, No. 9, pp. 1098-1101, September 1952.

[14 Rissanen, J.J. Generalized Kraft inequality and arithmetic coding.IBM 1. Res. Dev. 20 (May 1976), 198-203. Another early exposition of the idea of arithmetic coding.

[15] Pasco, R. (1976) "Source Coding Algorithms for Fast Data Compression," Ph. D. dissertation, Dept. of Electrical Engineering, Stanford University, Stanford, Calif

[16] Rissanen. J.J. Arithmetic codings as number representations. Acta Polytech. Stand. Math. 31 (Dec. 1979), 44-51. Further develops arithmetic coding as a practical technique for data representation.

[17] Rissanen, J., and Langdon, G.G. Arithmetic coding. IBM J. Res. Dev.23, 2 (Mar. 1979). 149-162. Describes a broad class of arithmetic codes.

[18] Langdon, G.G. An introduction to arithmetic coding. IBMI. Res. Dev. 28, 2 (Mar. 1984), 135-149. Introduction to arithmetic coding from the point of view of hardware implementation.

[19] A.Moffat and J.Katajainen, "In-place calculation of minimum-redundancy codes", 4th Intl. Workshop on Algorithms and Data Structures, Vol. 955, pp. 393-402, August 1995.

[20] J.Van Leeuwen, "On the construction of Huffman trees", 3rd International Colloquium on Automata, Languages and Programming, pp. 382-410, July 1976.

[21] M. Buro, "On the maximum length of Huffman codes", Information Processing Letters, Vol. 45, No.5, pp. 219-223, April 1993.

[22] H. C. Chen, Y. L. Wang and Y. F. Lan, "A memory efficient and fast Huffman decoding algorithm", Information Processing Letters, Vol. 69, No. 3, pp. 119-122, February 1999.

[23] R. Hashemian, "Direct Huffman coding and decoding using the table of code-lengths", Proc. International Conf. on Inform. Technology: Computers and Communications (ITCC '03), pp. 237-241, April 2003.

[24] S. Ho and P. Law, "Efficient hardware decoding method for modified Huffman code", Electronics Letters, Vol. 27, No. 10, pp. 855-856, May 1991.

[25] S. T. Klein, "Skeleton trees for the efficient decoding of Huffman encoded texts", Kluwer Journal of Inform. Retrieval, Vol. 3, No. 1, pp. 7-23, July 2000.

[26] L. L. Larmore and D. S. Hirschberg, "A fast algorithm for optimal length-limited Huffman codes", Journal of ACM, Vol. 37, No. 3, pp. 464-473, July 1999.

[27] A. Moffat and A. Turpin, "On the implementation of minimum-redundancy prefix codes", IEEE Trans. Commun., Vol. 45, No. 10, pp. 1200-1207, October 1997.

[28] O.Srinivasa Rao, S.Pallam Setty, "Efficient mapping methods of Elliptic Curve Crypto Systems" International Journal of Engineering Science and Technology, Vol. 2(8), 2010, pp. 3651-3656

[29] Vigila, S.; Muneeswaran, K.; "Implementation of text based cryptosystem using Elliptic Curve Cryptography", Advanced Computing, 2009. ICAC 2009. First International Conference on 13-15 Dec. 2009, Onpage(s): 82-85.

[30] Gupta, K.; Silakari, S.; Gupta, R.; Khan, S.A.; " An Ethical Way of Image Encryption Using ECC" Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on 23-25 July 2009,Onpage(s):342-345.

[31] R. Rajaram Ramasamy, M. Amutha Prabakar, M. Indra Devi, and M. Suguna, "Knapsack Based ECC Encryption and Decryption" International Journal of Network Security, Vol.9, No.3, PP.218–226, Nov. 2009

[32] O.Srinivasa Rao, S.Pallam Setty, "Comparative Study of Arithmetic and Huffman Data Compression Techniques for Koblitz Curve Cryptography" *International Journal of Computer Applications (0975 – 8887),Volume 14– No.5, January 2011*