Huffman Compression Technique in the Context of ECC for Enhancing the Security and Effective Utilization of Channel Bandwidth for Large Text

O.Srinivasa Rao
Dept. of CSE,
JNTUK University College of Engineering,
Vizianagaram, A.P.
India-535 003

Prof. S. Pallam Setty
CS & SE Dept.,
Andhra University College of Engineering,
Visakapatnam, A.P,
India-530 003

ABSTRACT

In this paper, we proposed a model for text encryption using elliptic curve cryptography (ECC) for secure transmission of large text and by incorporating the 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 Huffman compression technique in two ways i)x-y co-ordinates of encrypted text and ii) x-co-ordinates of the encrypted text. The resulting system saves the overall bandwidth and further enhances the security.

Keywords:

Elliptic Curve Cryptography (ECC), text encryption, Huffman 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.

1.2 Mathematical 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 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 data/message in the context of ECC for effective utilization of channel bandwidth .

2.1 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 [14, 15]. Since then there have been extensive researches on analysis, implementation issues and improvements of the Huffman coding theory in a variety of applications [16, 17, 18, 19, 20, 21and 22].

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 TEXT ENCRYPTION AND DECRYPTION WITH HUFFMAN

The proposed model at sender and receiver side for large 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 large text by aiming the effective utilization of channel bandwidth.

3.1 Encryption 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 $C_m = \{kG, P_m + kP_B\}$, 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 Huffman data compression which is then transmitted through in secured channel to the destination.

3.2 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 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 Text, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:
$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

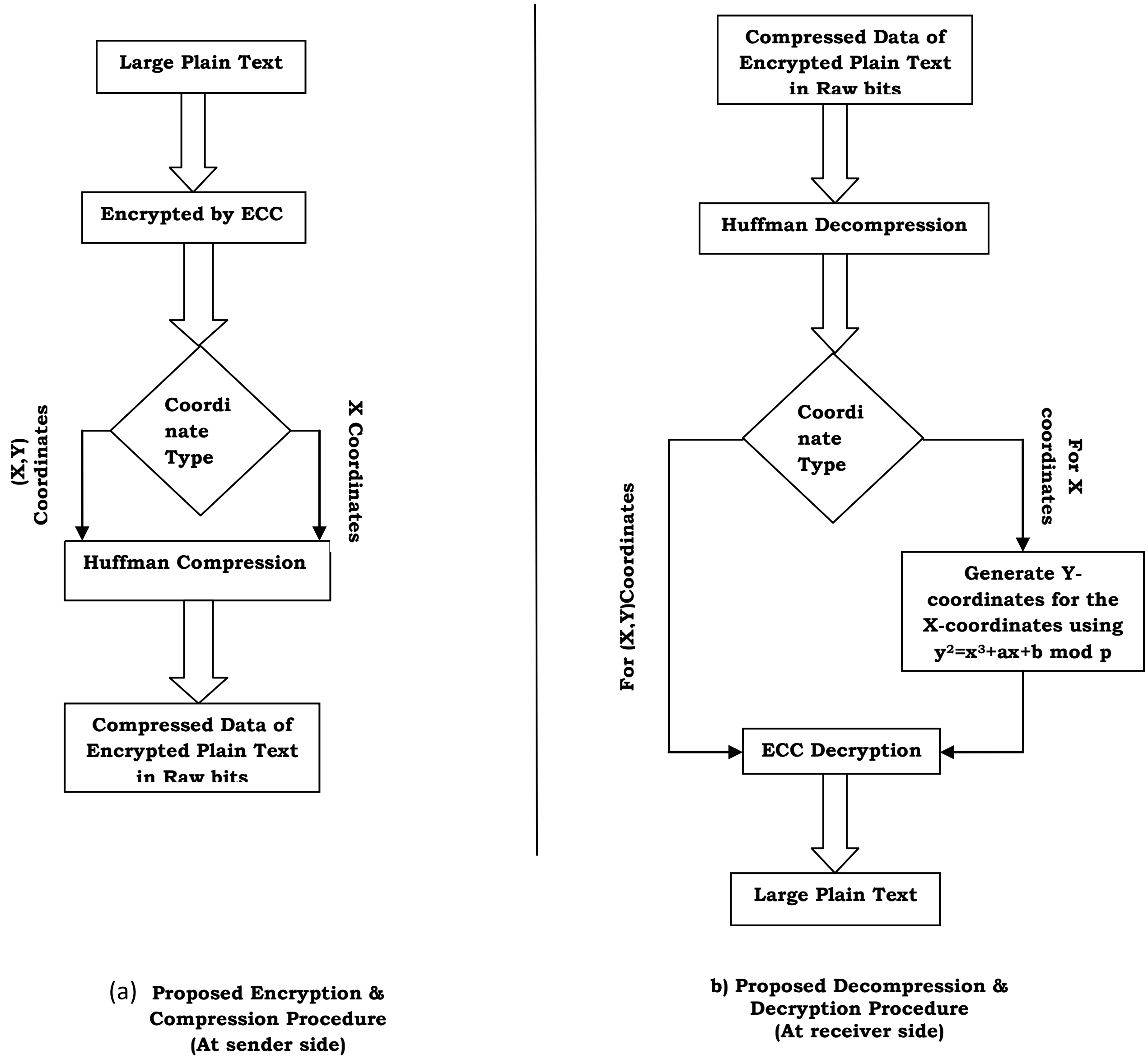

Figure 1 Proposed model at sender side and receiver side for Text in the context of ECC

For practical purpose, We have taken an elliptic curve $E_{571}(1,1)$ in the prime field and the alpha numerical characters are mapped [23,24] to the points of the EC. The mapped points are encrypted [25, 26] and computed compression ratio [27] for encrypted points using Huffman, from which we found the overall percentage of the bandwidth required and 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 $KG+n*[Pm + KP_B]$ 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, $KG+n(P_m + KP_B) \approx n(P_m + KP_B)$, hence the percentage of reduced bandwidth is give by,

$$\frac{\not{n}[P_m + KP_B]}{\not{n}[KG,(P_m + KP_B)]} = \frac{P_m + KP_B}{[KG,(P_m + KP_B)]}$$

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 $\approx n(P_m + KP_B)$, 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 1 to Table 4 and Figures 2 to Figures 3]. *In this case, the amount of bandwidth saved is 50% of original encrypted data + reduced size of the compressed data. Hence,*

$$OBWS\% = \frac{0.5 * Size\ of\ Encrypted\ Text + Compression\ Bits\ in\ (x,y)}{Size\ of\ Encrypted\ Text} * 100$$

The 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

$$OBWS\% = \frac{0.75 * Size\ of\ Encrypted\ Text + Compression\ Bits\ in\ (x,y)}{Size\ of\ Encrypted\ Text} * 100$$

The 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 (4.2).

4. DATA COMPRESSION TECHNIQUES FOR LARGE TEXT MESSAGE IN THE CONTEXT OF ECC

Arithmetic compression is limited to only small text messages so that for large text messages we analyzed the bandwidth requirements and saved in terms of Huffman Compression techniques only. The following experimental results show the Compression ratio, compression bits, percentage Bandwidth requirements and savings.

Table 1: Compressed Data and Compression Ratio in (*x, y*) Co-ordinates

Sl. No.	Input Text Files		OEDS	$EDS_{(x,y)}$	C	CR
	Size (kB)	Size (bits)				
1	1	8192	32768	16400	11233	0.6849390
2	2	16384	65536	32784	21940	0.6692288
3	3	24576	98304	49168	33659	0.6845712
4	4	32768	131072	65552	48573	0.7409842
5	5	40960	163840	81936	55645	0.6791276
6	6	49152	196608	98320	71225	0.724420
7	7	57344	229376	114704	85318	0.7438101
8	8	65536	262144	131088	93339	0.7120331
9	9	73728	294912	147472	95892	0.6502386
10	10	81920	327680	163856	120850	0.7375378

*OEDS Original Encrypted Data Size in bits
* C compression in bits
*CR Compression Ratio
$EDS_{(x,y)}$ Encrypted Data Size by considering both (*x, y*) co-ordinates

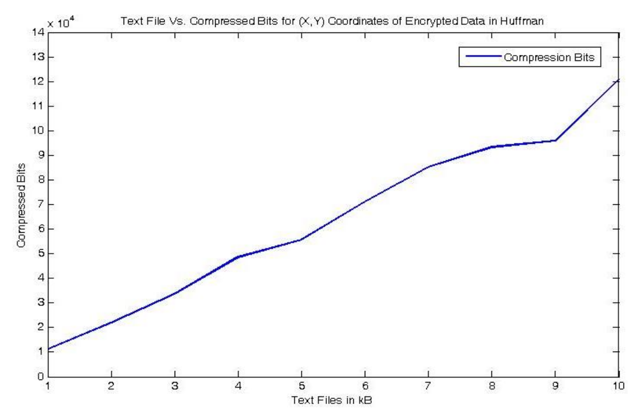

Figure 2: Text File Vs Compression bits in (*x, y*) co-ordinates of encrypted Data

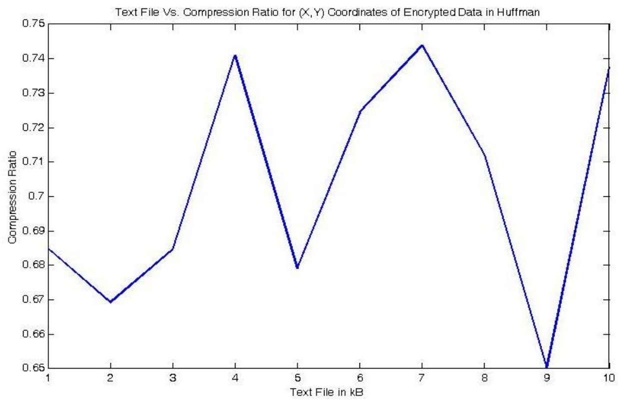

Figure 3: Text File Vs Compression Ratio in (x, y) co-ordinates of encrypted data

From the above Table 1, Figure 2 and Figure 3 one can observe that, the compressed bits vary from 11233 to 95892 where as compression ratio varies from 0.65 to 0.74

Table 2: O BWR % & OBWS % for (*x*, *y*)-Coordinates according to TBS

Sl. No.	Text Files		TBS	OBWR %	OBWS %
	Size (kB)	Size (bits)			
1	1	8192	21535	34.28039551	65.71960449
2	2	16384	43596	33.4777832	66.5222168
3	3	24576	64645	34.2397054	65.7602946
4	4	32768	82499	37.05825806	62.94174194
5	5	40960	108195	33.9630127	66.0369873
6	6	49152	125383	36.22690837	63.77309163
7	7	57344	144058	37.19569615	62.80430385
8	8	65536	168805	35.60600281	64.39399719
9	9	73728	199020	32.51546224	67.48453776
10	10	81920	206830	36.88049316	63.11950684

*TBS Total Bits Saved=OEDS-C

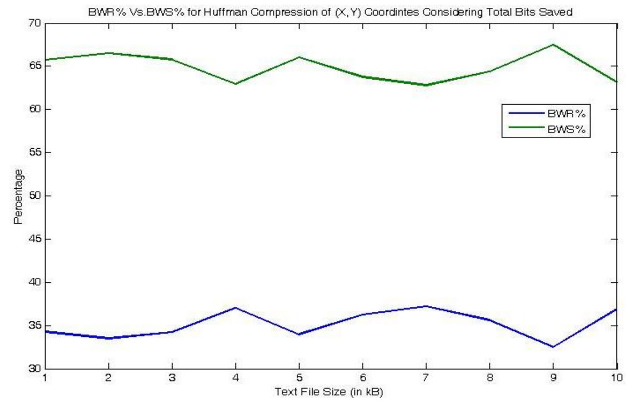

Figure 4: Text File Vs OBWR % and OBWS % in (*x*, *y*) co-ordinates of encrypted data

From the above Table 2 and Figure 5 one can observe that, the variation range in the overall percentage of bandwidth requirement and saving as follows:

S. No.	(x, y)	
	OBWR% Range	OBWS% Range
1	32.51- 37.19	62.8 – 67.48

Table 3: Compressed Data and Compression Ratio in (*x*)-Co-ordinates

Sl. No.	Text Files		OEDS	$EDS_{(x)}$	C	CR
	Size (kB)	Size (bits)				
1	1	8192	32768	8200	4736	0.577560976
2	2	16384	65536	16392	9752	0.594924353
3	3	24576	98304	24584	14340	0.583306215
4	4	32768	131072	32776	19111	0.583079082
5	5	40960	163840	40968	20651	0.504076352
6	6	49152	196608	49160	29218	0.594344996
7	7	57344	229376	57352	34097	0.594521551
8	8	65536	262144	65544	35811	0.5463658
9	9	73728	294912	73736	39150	0.530948248
10	10	81920	327680	81928	44983	0.549055268

$*EDS_{(x)}$ Encrypted Data by considering *x* co-ordinates

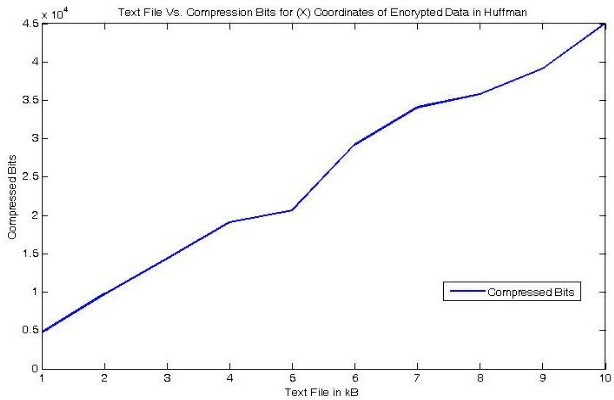

Figure 5: Text File Vs Compression bits in (x) co-ordinates of encrypted data

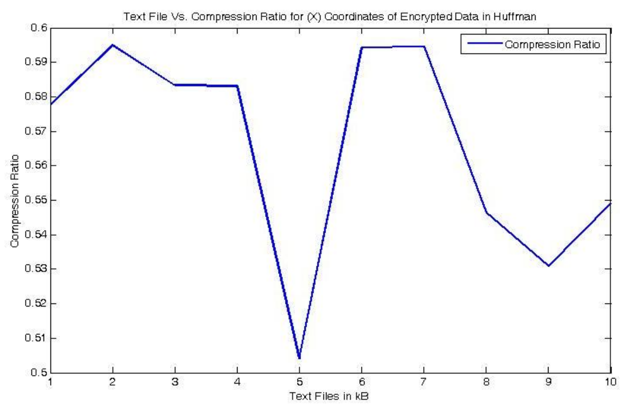

Figure 6: Text File Vs Compression Ratio for (x,y) co-ordinates of encrypted data

From the above Table 3, Figure 6 and Figure 7, one can observe that, the compressed bits varies from 4736 to 44983 where as compression ratio varies from 0.504 to 0.5949

Table 4: Overall BWR % & BWS % for (x)-Coordinates according to Total Bits Saved

Sl. No.	Text Files		TBS	OBWR %	OBWS %
	Size (kB)	Size (bits)			
1	1	8192	28032	14.453125	85.546875
2	2	16384	55784	14.88037109	85.11962891
3	3	24576	83964	14.58740234	85.41259766
4	4	32768	111961	14.58053589	85.41946411
5	5	40960	143189	12.60437012	87.39562988
6	6	49152	167390	14.86104329	85.13895671
7	7	57344	195279	14.8651123	85.1348877
8	8	65536	226333	13.66081238	86.33918762
9	9	73728	255762	13.27514648	86.72485352
10	10	81920	282697	13.72772217	86.27227783

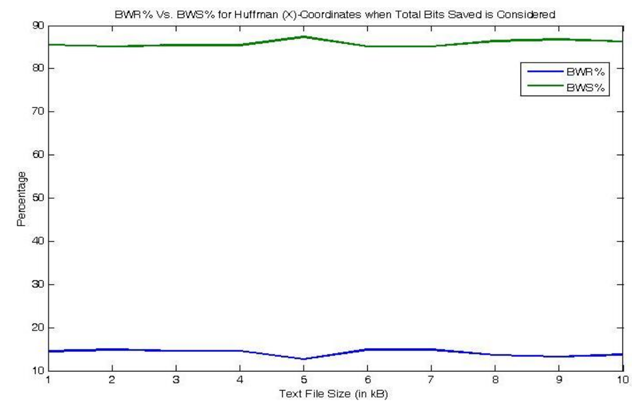

Figure 7: Text File Vs OBWR % and OBWS % in (*x, y*) co-ordinates of encrypted data

From the above Table 4 and Figure 8 one can observe that, the variation range in the overall percentage of bandwidth requirement and saving as follows:

S.No.	*(x)* co-ordinate	
	OBWR% Range	OBWS% Range
1	12.6- 14.88	85.11 – 87.39

Table:5: Comparison of OBWR% and OBWS% in (x, y)Vs x co-ordinates:

Sl. No.	Text Files		OBWR %		OBWS%	
	Size (kB)	Size (bits)	*(x, y)*	*(x)*	*(x, y)*	*(x)*
1	1	8192	34.280395	14.4531	65.719604	85.5468
2	2	16384	33.47778	14.880371	66.52221	85.119628
3	3	24576	34.23970	14.587402	65.76029	85.412597
4	4	32768	37.058258	14.580535	62.941741	85.419464
5	5	40960	33.96301	12.604370	66.03698	87.395629
6	6	49152	36.226908	14.861043	63.773091	85.138956
7	7	57344	37.195696	14.86511	62.804303	85.13488
8	8	65536	35.606002	13.660812	64.393997	86.339187
9	9	73728	32.515462	13.275146	67.484537	86.724853
10	10	81920	36.880493	13.727722	63.119506	86.272277

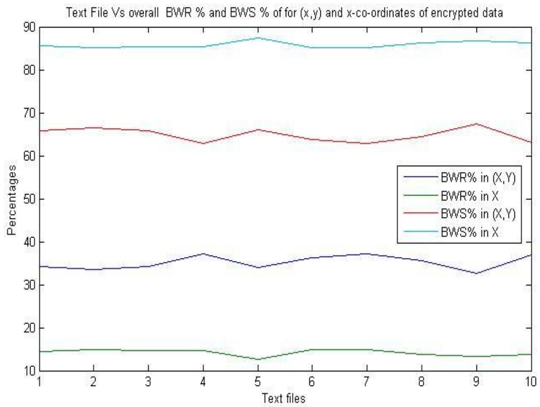

Figure 8: comparison of OBWR % and OBWS % in (*x*, *y*) and *x* co-ordinates of encrypted data for given text files

From the above Table 5 and Figure 9 one can observe that, the variation range in the overall percentage of bandwidth requirement and saving in (x,y) and x co-ordinates as follows.

Sl. No.	OBWR% Range		OBWS% Range	
	(x,y)	*(x)*	*(x,y)*	*(x)*
1	32.51- 37.19	12.6- 14.88	62.8 – 67.48	85.11 – 87.39

5. THE COMPRESSION RATIO, THE OBWR% and OBWS% ARE COMPUTED FOR TEXT SIZES OF 10 KB 100 KB IN STEPS OF 10KB

Table 6: Compressed Data and Compression Ratio for (*x*, *y*)-Coordinates

Sl. No.	Text Files		OEDS	$EDS_{(x,y)}$	C	CR
	Size (kB)	Size (bits)				
1	10	81920	327680	163856	120850	0.7375378
2	20	163840	655360	327696	219944	0.6711830
3	30	245760	983040	491536	350005	0.7120638
4	40	327680	1310720	655376	470652	0.7181404
5	50	409600	1638400	819216	544181	0.6642704
6	60	491520	1966080	983056	635361	0.6463121
7	70	573440	2293760	1146896	791565	0.6901802
8	80	655360	2621440	1310736	911439	0.6953642
9	90	737280	2949120	1474576	945410	0.6411402
10	100	819200	3276800	1638416	1058175	0.6458524

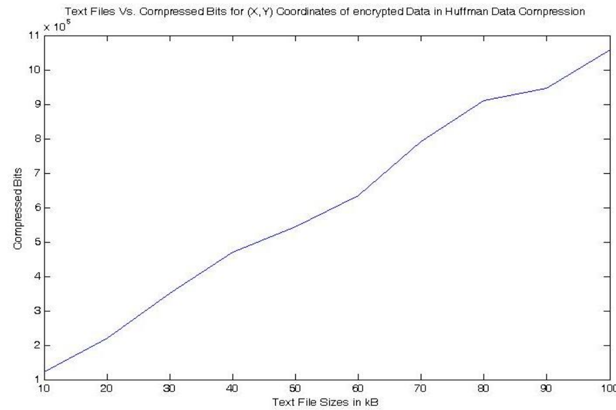

Figure 9: Text File Vs Compression bits for (*x*, *y*) co-ordinates of encrypted data

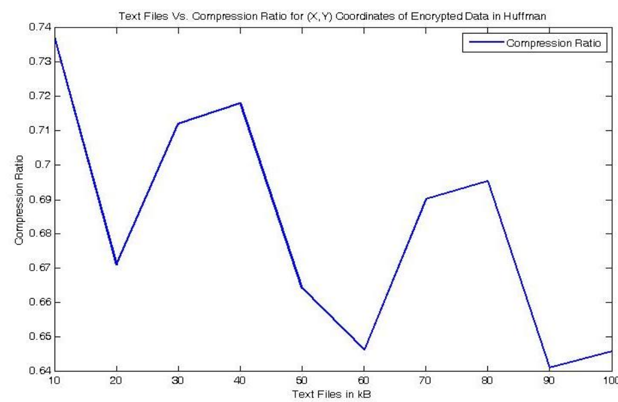

Figure 10: Text File Vs Compression Ratio for (*x*, *y*) co-ordinates of encrypted data

From the above Table 6, Figure 10 and Figure 11, one can observe that, the compressed bits varies from 120850 to 1058175 where as compression ratio varies from 0.64 to 0.7375

Table 7: O BWR % & OBWS % in (*x*, *y*) Co-ordinates according to TBS

Sl. No.	Text Files		TBS	OBWR %	OBWS %
	Size (kB)	Size (bits)			
1	10	81920	206830	36.88049316	63.11950684
2	20	163840	435416	33.56079102	66.43920898
3	30	245760	633035	35.60434977	64.39565023
4	40	327680	840068	35.90789795	64.09210205
5	50	409600	1094219	33.21417236	66.78582764
6	60	491520	1330719	32.31613159	67.68386841
7	70	573440	1502195	34.50949533	65.49050467
8	80	655360	1710001	34.76863861	65.23136139

9	90	737280	2003710	32.05735948	67.94264052
10	100	819200	2218625	32.29293823	67.70706177

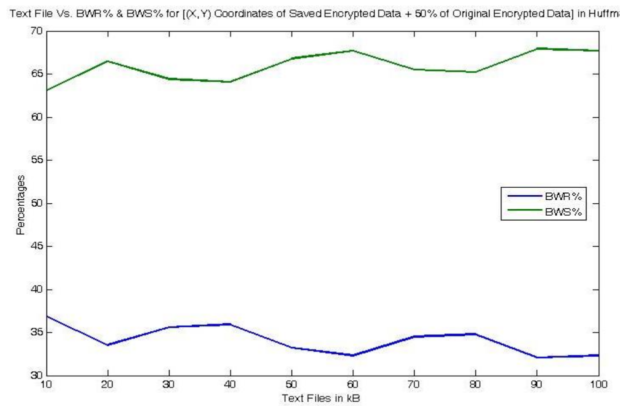

Figure 11: Comparison of OBWR % and OBWS % in (*x*, *y*) co-ordinates of encrypted data for the given text

From the above Table 7 and Figure 12 one can observe that, the variation range in the overall percentage of bandwidth requirement and saving in (x,y) co-ordinates as follows.

S.No.	*(x,y)* co-ordinates	
	OBWR% Range	OBWS% Range
1	32.05- 36.88	63.11- 67.94

Table 8: Compressed Data and Compression Ratio for (X)-Coordinates

Sl. No.	Text Files		OEDS	$EDS_{(x)}$	C	CR
	Size (kB)	Size (bits)				
1	10	81920	327680	81928	44983	0.5490552
2	20	163840	655360	163848	95308	0.5816854
3	30	245760	983040	245768	138196	0.5623026
4	40	327680	1310720	327688	183207	0.5590897
5	50	409600	1638400	409608	228929	0.5588977
6	60	491520	1966080	491528	265448	0.5400465
7	70	573440	2293760	573448	321611	0.5608372
8	80	655360	2621440	655368	363576	0.5547661
9	90	737280	2949120	737288	399123	0.5413393
10	100	819200	3276800	819208	459921	0.5614215

Figure 12: Text File Vs Compression bits for (x) co-ordinates of encrypted data

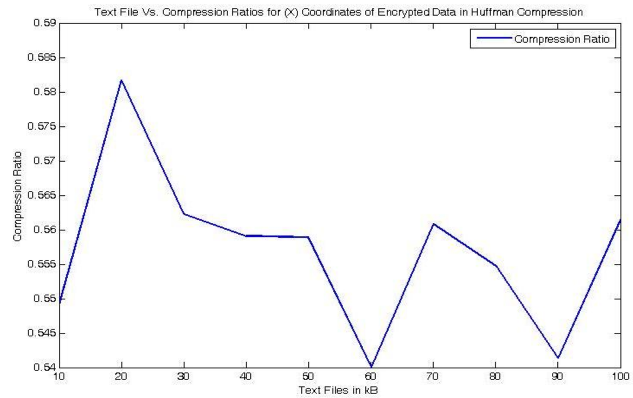

Figure 13: Text File Vs Compression Ratio for (*x*, *y*) co-ordinates of encrypted data

From the above Table 8, Figure 13 and Figure 14, one can observe that, the compressed bits vary from 44983 to 459921 where as compression ratio varies from 0.54 to 0.58

Table 9: OBWR % & OBWS % in (*x*) Co-ordinates according to TBS

Sl. No.	Text Files		TBS	OBWR %	OBWS %
	Size (kB)	Size (bits)			
1	10	81920	282697	13.72772217	86.27227783
2	20	163840	560052	14.54284668	85.45715332
3	30	245760	844844	14.05802409	85.94197591
4	40	327680	1127513	13.97758484	86.02241516
5	50	409600	1409471	13.97271729	86.02728271
6	60	491520	1700632	13.50138346	86.49861654
7	70	573440	1972149	14.02112688	85.97887312
8	80	655360	2257864	13.86932373	86.13067627
9	90	737280	2549997	13.53363037	86.46636963
10	100	819200	2816879	14.03567505	85.96432495

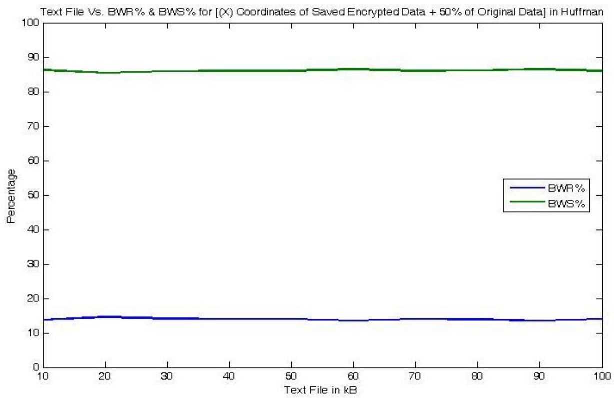

Figure 14: Comparision of OBWR % andOBWS % in (x,y) co-ordinates of encrypted data for given text files

From the above Table 9 and Figure 15 one can observe that, the variation range in the overall percentage of bandwidth requirement and saving in *x* co-ordinates as follows.

S.No.	*(x)* co-ordinates	
	OBWR% Range	**OBWS% Range**
1	13.5- 14.54	85.45- 86.49

Table 10: Comparison of OBWR% and OBWS % in *(x,y)* Vs *(x)* co-ordinates:

Sl. No.	Text Files		OBWR %		OBWS%	
	Size (kB)	**Size (bits)**	**(x, y)**	**(x)**	**(x, y)**	**(x)**
1	10	81920	36.880493	13.727722	65.719604	86.272277
2	20	163840	33.560791	14.542846	66.52221	85.457153
3	30	245760	35.604349	14.058024	65.76029	85.941975
4	40	327680	35.907897	13.977584	62.941741	86.022415
5	50	409600	33.214172	13.972717	66.03698	86.027282
6	60	491520	32.316131	13.501383	63.773091	86.498616
7	70	573440	34.509495	14.021126	62.804303	85.978873
8	80	655360	34.768638	13.869323	64.393997	86.130676
9	90	737280	32.057359	13.533630	67.484537	86.466369
10	100	819200	32.292938	14.035675	63.119506	85.964324

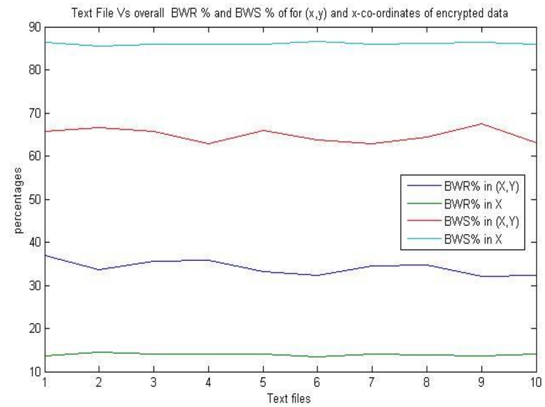

Figure 15: comparison of OBWR % and OBWS % in (*x*, *y*) and *x*-co-ordinates of encrypted data for the given text files

From the above Table 10 and Figure 16 one can observe that, the variation range in the overall percentage of bandwidth requirement and saving in (*x, y*) and *x* co-ordinates as follows.

S.No.	OBWR% Range		OBWS% Range	
	(x, y)	***(x)***	***(x, y)***	***(x)***
1	32.05- 36.88	13.5- 14.54	63.11- 67.94	85.45- 86.49

6. CONCLUSION

The experiments are conducted for the following cases, by considering only *x* co-ordinate and both *(x, y)* co-ordinates of the different encrypted text for transmission in Huffman compression.

For large text, the experiments are conducted for the following cases, by considering only *x* co-ordinate and both *(x, y)* co-ordinates of the encrypted large text of the size varying from 1 kB to 10 kB in steps of 1kB and from 10 kB to 100 kB in steps of 10 kB for transmission in Huffman compression:

Irrespective of the case, when both (x,y) coordinates, are considered for transmission, the overall percentage of bandwidth requirement (OBWR %)varies from 32.05% to 37.19% and the percentage of Bandwidth Saving (OBWS %) varies from 62.8% to 67.94%.

When only *x* co-ordinate for transmission is considered, the overall percentage of bandwidth requirement (OBWR %) varies from 12.6% to 14.88% and the percentage of Bandwidth Saving (O*BWS %*) varies from 85.11% to 87.39%.Hence it is concluded that by incorporating the Huffman compression to ECC not only enhances the security but also enhances the utilization of the channel bandwidth also.

7. 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 Techniques Based 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] 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.

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

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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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

[24] 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, on page(s): 82-85.

[25] 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.

[26] 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

[27] 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