Cryptography by Karatsuba Multiplier with ASCII Codes

Tarun Narayan Shankar
Dept. of Computer Sc & Engg.
GMR Institute of Technology,RAJAM,
SRIKAKULAM, AP,INDIA, PIN-532127

G.Sahoo
Dept.of Information Technology & MCA
Birla Institute of Technology,Mesra,Ranchi
RANCHI-835215, INDIA

ABSTRACT
In this paper we describe Cryptography by using Karatsuba multipliers and ASCII codes implementing through coordinate geometry for data encryption and decryption with its code in matlab. Elliptic curve cryptography is an asymmetric key cryptography. It includes (i) public key generation on the elliptic curve and its declaration for data encryption and (ii) private key generation and its use in data decryption depended on the points on two dimensional elliptical curve. We also discuss the implementation of ECC on binary field. An overview of ECC implementation on two dimensional representation of ASCII codes with coordinate systems and data encryption through Elgamal Encryption technique has been discussed. Karatsuba multiplier is a fast process to solve the Elliptic curve cryptography problems. Here we have applied Karatsuba multiplier for point multiplication. Much attention has been given on the mathematical implementation of elliptic curves through Karatsuba multiplier. For cryptographic purposes, specifically results of the group formed by an elliptic curve over a finite field $E(F_2{}^m)$, and showing how this can form public key cryptographic systems for use in both encryption and key exchange. Finally we describe how to encrypt and decrypt the data with the ASCII codes through Karatsuba multiplier and its implimentation through matlab.

Categories and Subject Descriptors
E.3.[Data Encryption]: *Public key cryptosystems,Elliptic curve cryptography,Karatsuba multiplication,ASCII table.*

General Terms:*Algorithm,Security,field*

Keywords:*Cryptography,Field,Encryption,Decryption, ASCII.*

1.INTRODUCTION
In ECC a 160-bit key provides the same security as compared to the traditional crypto system RSA[6] with a 1024-bit key, thus lowers the computer power. Therefore,ECC offers considerably greater security for a given key size.Consequently, a key with smaller size makes it possible a much more compact implementations for a given level of security,

which means faster cryptographic operations, running on smaller chips or more compact software. After using Karatsuba multiplier (multiplication and addition) [2]fastest cryptographic operational speed should be gained. Further, there are extremely efficient, compact hardware implementations are available for ECC exponentiation operations, offering potential reductions in implementation footprint even beyond those due to the smaller key length alone. Elliptic curve cryptography is not only emerged as an attractive public key crypto-system for mobile/wireless environments but also provides bandwidth savings. The use of elliptic curve in cryptography was proposed by Miller and Koblitz. Elliptic curve cryptography is not easy to understand by attacker. So not easy to break. The choice of the type of elliptic curve is dependent on its domain parameters, the finite field representation, elliptic curve algorithms for field arithmetic[4] as well as elliptic curve arithmetic. The optimum selection of these parameters also depends on the security conditions under consideration. There are several research papers in this subject available in the literature covering different areas like hardware and software implementations. In the above respect it can be mentioned here that one can define encryption points as $\underline{e}_i$ and $\underline{e}_j$ by a specified algorithm but it is not yet possible for the case of plain text. In this paper we have discussed about the encryption and decryption with Elliptical curves $E(F_2{}^m)$ and an attempt has been made to represent plaintext in two dimensional form with the help of an ASCII code table so that Elgamal encryption technique[9] can be used for the said ECC. Karatsuba multiplication methods have been used for less complexity and fast process. It can be mentioned here that ECC produces both private key and public key. Private key is known as secret key. In symmetric key cryptography single key uses for both encryption and decryption. In asymmetric key algorithm it however uses only for decryption of encrypted message. In asymmetric key cryptography, public key is used for message encryption and widely distributed for public. Elliptic curve cryptography is asymmetric key cryptography by nature. For the completeness of the paper, the description and use of the elliptic curves with Karatsuba multiplier[2][3] and ASCII codes is given in few of the subsequent section. In section 6 we describe the methodology of encryption for plaintext followed by conclusion in the last section and in section 7 coding in matlab.

				b_4	0	0	0	0	0	0	0	0
				b_3	0	0	0	0	1	1	1	1
				b_2	0	0	1	1	0	0	1	1
				b_1	0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	‘	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	“	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	EQN	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	‘	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O		o	DEL

(Fig-1) (ASCII table for 128 characters)

2.1ASCII codes

Before ASCII was developed, the encodings in use included 26 alphabetic characters, 10 numerical digits, and from 11 to 25 special graphic symbols. More than 64 codes were required in ASCII. ASCII codes (Fig-1) represent as text in computers, communications equipment, and other devices that work with text. Most modern character encodings which support many more characters than did the original have a historical basis in ASCII. ASCII[12] developed from telegraphic codes and its first commercial use was as a seven-bit tele printer code promoted by Bell data services. Work on ASCII formally began October 6, 1960 with the first meeting of the ASA X3.2 subcommittee. The first edition of the standard was published in 1963, a major revision in 1967, and the most recent update in 1986. Compared to earlier telegraph codes, the proposed Bell code and ASCII were both ordered for more convenient sorting (i.e., alphabetization) of lists, and added features for devices other than teleprinters. Some ASCII features, including the "ESCape sequence", were due to Robert Bemer. ASCII includes definitions for 128 characters: 33 are non-printing, mostly obsolete control characters that affect how text is processed; 94 are printable characters (the space is not printable). The ASCII character encoding or a compatible extension is used on nearly all common computers, especially personal computers and workstations.

The representation of each and every character is with the seven bits (b_7 to b_1),e.g. the representation of A is (1000001)=65,similarly other characters are coded like this.

2.2. ASCII table for two dimensional coordinate representation:

Input and output devices that communicate with people and the computer are usually involved in the transfer of alphanumeric information to and from the device and the computer.It uses seven bits to code 128 characters shown as the above table. In case of ASCII code cryptography the arrangement of the characters are distinct from the table(Fig-1.2).According to this each and every ASCII code have two dimensional coordinate representation[10][11] with eight bits .That is (b_4 to b_1) for X-coordinate and (b_4 to b_1) for Y-coordinate. Now the representation of B is (0010,0100),and for encryption we can add this with the point which is lying on the elliptic curve. Similarly we can represent another characters. It’s clear that coordinate representation is different than ASCII representation. With these representation characters can be taken different points of elliptic curve for data encryption with the following algorithms in section-6.

3.1Elliptic Curve Arithmetic

Elliptic curve cryptography is based on binary field arithmetic[1]. Note that elliptic curves are not ellipses. They are so named because of the fact that ellipses are formed by quadratic curves. Elliptic curves are always cubic and have a relationship to elliptic integrals in mathematics[4][5] where the elliptic integral can be used to determine the arc length of an ellipse. An elliptic curve in its "standard form" is described by

$$y^2 = x^3 + ax + b \qquad \text{.....(1.1)}$$

For the polynomial $x^3 + ax + b$, the discriminant can be given as

$$D = -(4a^3 + 27b^2) \qquad \text{.....(1.2)}$$

This discriminant must not become zero for an elliptic curve polynomial $x^3 + ax + b$ to possess three distinct roots.If the discriminant is zero, that would imply that two or more roots have coalesced, giving the curves in singular form. It is not safe to use singular curves for cryptography as they are easy to crack. Due to this reason we generally take non-singular curves for data encryption.

Let $P(x_p,y_p)$ and $Q(x_q,y_q)$ be the two points on the curve of Eq.(1.1).Then the point additions $\mathcal{P} + \mathcal{Q}$, as well as point doubling $\mathcal{P} + \mathcal{P}$ are two operations defined on elliptic curve E which can geometrically be represented by tangent and chord operation. By applying the point addition and doubling operation we can multiply a scalar k with a point $\mathcal{P}$, such that $k\mathcal{P} = \mathcal{Q}$, where k is a scalar. Given $\mathcal{P}$ and $\mathcal{Q}$, it is computationally infeasible to obtain k. If k is sufficiently large, k is the discrete logarithm of $\mathcal{Q}$ to the base $\mathcal{P}$. Hence the main operation involved in ECC is related to the point multiplication i.e. multiplication of a scalar k with any point $\mathcal{P}$ on the curve to obtain another point $\mathcal{Q}$ on the curve.

3.2.Elliptic Curves over F_{2^m} :

What makes the binary finite fields more convenient for hardware implementations is that the elements of $GF(2^m)$ can be represented by m-bit binary code words. The addition operation in $GF(2^m)$ is like the XOR operation on bit fields.

That is $x + x = 0$ for all $x \in GF(2^m)$.

This implies that a finite field of form $GF(2^m)$ is of characteristic 2. The equation of the elliptic curve on a binary field F_2^m is

$y^2 + xy = x^3 + ax^2 + b$, where $b \neq 0$.

Here the elements of the finite field are integers of length at most m bits. These numbers can be considered as a binary polynomial of degree m – 1.In binary polynomial the coefficients can only be 0 or 1.The addition of two points on a curve over is F_2^m defined as

$$(x_1,y_1) + (x_2,y_2) = (x_3,y_3)$$

Where

$$(x_3,y_3) = (\alpha^2 + \alpha + x_1 + x_2 + a\ ,\ \alpha(x_1+x_3)+x_3+y_1) \qquad \text{.....(1.3)}$$

Where

$$\alpha = (y_1+y_2)/(x_1+x_2)$$

4.1.Polynomial Arithmetic :

Elliptic curve over field F_2^m [7]involves arithmetic of integer of length m bits. These numbers can be considered as binary polynomial of degree m – 1. The binary string $(a_{m-1} \ldots a_1 a_0)$ can be expressed as polynomial

$$A(x) = \sum_{i=0}^{m-1} a_i x^i = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_2x^2 + a_1x + a_0$$

where $a_i = 0$ or 1.

For e.g., a 4 bit number 1101_2 can be represented by polynomial as $x^3 + x^2 + 1$.Similar to the modulus p on modular arithmetic, there is an irreducible polynomial of degree m in polynomial arithmetic. If in any operation the degree of polynomial is greater than or equal to m, the result is reduced to a degree less than m using irreducible polynomial also called as reduction polynomial. In binary polynomial the coefficients of the polynomial can be either 0 or 1. If in any operation the coefficient becomes greater than 1, it can be reduced to 0 or 1 by modulo 2 operation on the coefficient. All the operations below are defined in field F_2^4 are on irreducible polynomial $f(x) = x^4 + x + 1$. Since m = 4 the operation involves polynomial of degree 3 or lesser.

4.2Irreducible polynomial

A polynomial [5][8] f(x) is known as to be irreducible if we can't write f(x)=h(x).g(x) .For any polynomials h(x) ,g(x) of degree strictly less than the degree of f(x) . An irreducible polynomial of degree m over F_2^m should satisfy these necessary conditions.

*The constant term $a_0 = 1$

*There is an odd number (≥3)of nonzero terms , otherwise ,f(x) whose number of nonzero terms is even has a factor (x+1).

*There must be at least one term of odd degree otherwise ,f(x) of all even powers is a square of a polynomial of degree(m/2).If in any polynomial arithmetic operation the resultant polynomial is having degree greater than or equal to m,
it is reduced to a polynomial of degree less than m by the irreducible polynomial .NIST recommended curve $m \in \{113,131,163,193,233,239,283,409,571\}$ with the following irreducible functions.

F_2113 $f(x) = x^{113} + x^9 + 1$

F_2131 $f(x) = x^{131} + x^8 + x^3 + x^2 + 1$

F_2163 $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$

F_2193 $f(x) = x^{193} + x^{15} + 1$

F_2233 $f(x) = x^{233} + x^{74} + 1$

F_2239 $f(x) = x^{239} + x^{36} + 1$

F_2283 $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$

F_2409 $f(x) = x^{409} + x^{87} + 1$

F_2571 $f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

Any irreducible polynomial can be taken for **C**ryptography By **K**aratsuba **M**ultiplier and **A**SCII **C**odes according to its key length.

5.Karatsuba-Ofman Method:

Karatsuba-Ofman's algorithm[2][3]is considered one of the fastest ways to multiply long integers. Karatsuba-Ofman's algorithm is based on a divide-and-conquer strategy. A multiplication of a 2*n*-digit integer is reduced to two *n*-digits multiplications, one(*n*+1)-digits multiplication, two *n*-digits subtractions, two left-shift operations, two *n*-digits additions and two 2*n*-digits additions.
Let *X* and *Y* be the binary representation of two long integers:

$$X = \sum_{i=0}^{k-1} x_i 2^i \quad \text{and} \quad Y = \sum_{i=0}^{k-1} y_i 2^i$$

Now compute the product of X and Y. For this the operands X can be divided into two parts X^H, and X^L .Similarly Y can be divided into two parts Y^H ,and Y^L. Let k = 2n then

$$X = 2^n \left[\sum_{i=0}^{n-1} x_{i+n} 2^i \right] + \sum_{i=0}^{n-1} x_i 2^i = X^H 2^n + X^L$$

$$Y = 2^n \left[\sum_{i=0}^{n-1} y_{i+n} 2^i \right] + \sum_{i=0}^{n-1} y_i 2^i = Y^H 2^n + Y^L$$

The product is computed as
$P = (X^H 2^n + X^L)(Y^H 2^n + Y^L) = 2^{2n}(X^H Y^H) + 2^n(X^H Y^L + X^L Y^H) + X^L Y^L = 2^{2n} X^H Y^H + X^L Y^L + ((X^H + X^L)(Y^H + Y^L) - X^H Y^H - X^L Y^L) 2^n$

If $P1 = X^H Y^H$
$P2 = X^L Y^L$
and $P3 = (X^H + X^L)(Y^H + Y^L)$ then the product is
$P = 2^{2n} P1 + P2 + 2^n (P3-P1-P2)$

The Karatsuba Ofman's algorithm is based on 2n bits multiplication can be reduced to three n bits multiplications as P1,P2,P3.Where the function size *Size*(X) returns the number of bits of X. The function H(X) returns the higher part of the X. Function L(X) returns the lower half of the X. Rightshift(X,n) returns $X2^n$ and one bit multiplication(X,Y) returns XY when both X and Y are formed by a single bit.

Algorithm:

Step1:P = M(X,Y)

If (Size(X) = 1) **Then** M= One Bit Multiplier(X, Y)

Else

Step-2:P1 := MUL(H(X), H(Y));

Step3: P2 := MUL(L (X), L (Y));

Step4: P3 := MUL(H (X)+L(X), H(Y)+L(Y));

Step5: P := Right Shift (P1, Size(X)) + Right Shift (P3-P1-P2,Size(X)/2) + P2

Endif

Step6: Reduce P with irreducible polynomial

6.Algorithm for encryption and decryption:

Algoritm 1:

Step 1. Use an appropriate data structure to store the text to be encrypted.

Step 2. Read the table in row-major form and find the character stored in that position.

Step 3. Note the row and column values.
Step 4. Assign these values to the same character in all positions it appears.

Now, we define an analogous algorithm due to ElGamal[10] for encprypting the required text as follows:

Algorithm 2:

Step1.Select E(a,b)with an elliptic curve over GF(*p*) or $GF(2^m)$.
Step2. Select a point on the curve $e_i = (x_i, y_i)$.

Step3. Select g
Step4. Calculate $e_j = (x_j, y_j) = g * e_i$
Step5. Announce e_i ,e_j as public key and keep *"g"* as a private key.

Step6. {Encryption}

Now select *h* a number in plaintext P with the coordinate from the Table and calculate pair of points on the text as ciphertext.

Step7. $C_i = h * e_i$

$C_j = (xp_i, yp_j) + h * e_j$

Step8. {Decryption}
After receiving ciphertext C_i and C_j calculate

P(plain text) with the private key *g* .
$(xp_i, yp_j) = C_j - (g * C_i)$ [Here the (-)sign means adding with inverse.]

Step9. Read the characters from the co-ordinates(xp_i, yp_j)

```
ASCIItable = [{'NUL'},{[0 0]};
 {'SOH'},{[1 0]};
{'STX'},{[2 0]};
 {'ETX'},{[3 0]};
 {'EOT'},{[4 0]};
 {'EQN'},{[5 0]};
 {'ACK'},{[6 0]};
 {'BEL'},{[7 0]};
```

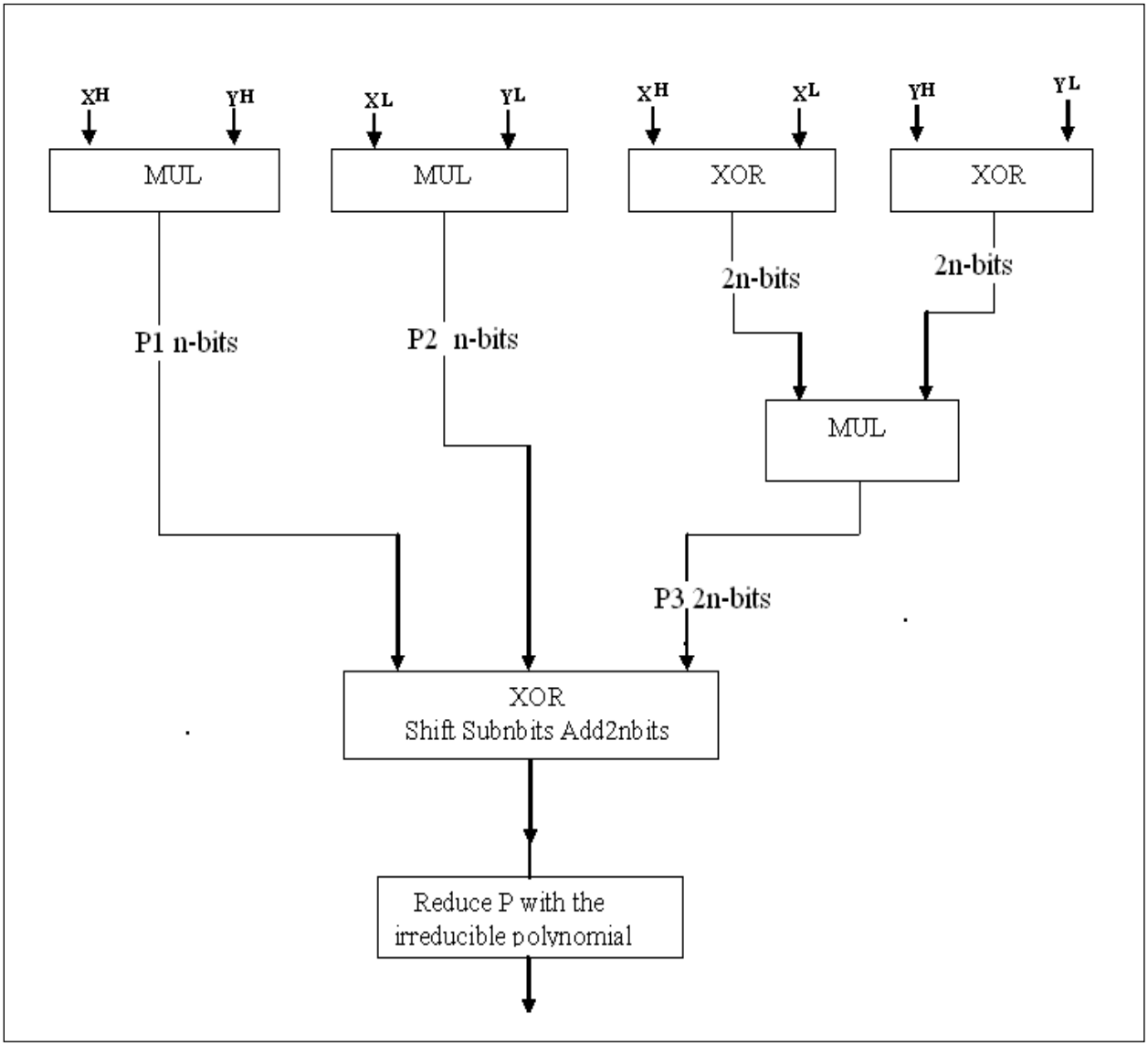

Fig-2 Hardware diagram for Karatsuba multiplication

7.Matlab coding for Cryptography by ASCII code with Karatsuba multiplier

7.1The Matlab code for ASCII Table

Below is the Matlab code for ASCII table.ASCII.m

```
{'BS'} ,{[8 0]};
{'HT'} ,{[9 0]};
{'FF'},{[12 0]};
 {'CR'},{[13 0]};
 {'SO'},{[14 0]};
 {'SI'},{[15 0]};
 {'DLE'},{[0 1]};
 {'DC1'},{[1 1]};
 {'DC2'},{[2 1]};
 {'DC3'},{[3 1]};
 {'DC4'},{[4 1]};
 {'NAK'},{[5 1]};
 {'SYN'},{[6 1]};
 {'ETB'},{[7 1]};
 {'CAN'},{[8 1]};
 {'EM'},{[9 1]};
 {'SUB'},{[10 1]};
 {'ESC'},{[11 1]};
 {'FS'},{[12 1]};
 {'GS'},{[13 1]};
 {'RS'},{[14 1]};
 {'US'},{[15 1]};
 {'SP'},{[0 2]};
 {'!'},{[1 2]};
 {'"'},{[2 2]};
 {'#'},{[3 2]};
 {'$'},{[4 2]};
 {'%'},{[5 2]};
 {'&'},{[6 2]};
 {'`'},{[7 2]};
 {'('},{[8 2]};
 {')'},{[9 2]};
 {'*'},{[10 2]};
 {'+'},{[11 2]};
 {','},{[12 2]};
 {'-'},{[13 2]};
 {'.'},{[14 2]};
 {'/'},{[15 2]};
 {'0'},{[0 3]};
 {'1'},{[1 3]};
 {'2'},{[2 3]};
 {'3'},{[3 3]};
 {'4'},{[4 3]};
 {'5'},{[5 3]};
 {'6'},{[6 3]};
 {'7'},{[7 3]};
 {'8'},{[8 3]};
 {'9'},{[9 3]};
 {':'},{[10 3]};
 {';'},{[11 3]};
 {'<'},{[12 3]};
 {'='},{[13 3]};
 {'>'},{[14 3]};
 {'?'},{[15 3]};
 {'@'},{[0 4]};
 {'A'},{[1 4]};
 {'B'},{[2 4]};
 {'C'},{[3 4]};
 {'D'},{[4 4]};
 {'E'},{[5 4]};
 {'F'},{[6 4]};
 {'G'},{[7 4]};
 {'H'},{[8 4]};
 {'I'},{[9 4]};
 {'J'},{[10 4]};
 {'K'},{[11 4]};
 {'L'},{[12 4]};
 {'M'},{[13 4]};
 {'N'},{[14 4]};
 {'O'},{[15 4]};
 {'P'},{[0 5]};
 {'Q'},{[1 5]};
 {'R'},{[2 5]};
 {'S'},{[3 5]};
 {'T'},{[4 5]};
 {'U'},{[5 5]};
 {'V'},{[6 5]};
 {'W'},{[7 5]};
 {'X'},{[8 5]};
 {'Y'},{[9 5]};
 {'Z'},{[10 5]};
 {'['},{[11 5]};
 {'\'},{[12 5]};
 {']'},{[13 5]};
 {'^'},{[14 5]};
 {'_'},{[15 5]};
 {'`'},{[0 6]};
 {'a'},{[1 6]};
 {'b'},{[2 6]};
 {'c'},{[3 6]};
 {'d'},{[4 6]};
 {'e'},{[5 6]};
 {'f'},{[6 6]};
 {'g'},{[7 6]};
 {'h'},{[8 6]};
 {'i'},{[9 6]};
 {'j'},{[10 6]};
 {'k'},{[11 6]};
 {'l'},{[12 6]};
 {'m'},{[13 6]};
 {'n'},{[14 6]};
 {'o'},{[15 6]};
 {'p'},{[0 7]};
 {'q'},{[1 7]};
 {'r'},{[2 7]};
 {'s'},{[3 7]};
 {'t'},{[4 7]};
 {'u'},{[5 7]};
 {'v'},{[6 7]};
 {'w'},{[7 7]};
 {'x'},{[8 7]};
 {'y'},{[9 7]};
 {'z'},{[10 7]};
 {'{'},{[11 7]};
 {'|'},{[12 7]};
 {'}'},{[13 7]};
 {'~'},{[14 7]};
 {'DEL'},{[15 7]}];
```

7.2 The Matlab code for Karatsubamultiplication. Karatsuba.m

```
function r = karatsuba(p,q)
n = length(p)-1;
```

```
if(n==0)
   r=p*q;
   return;
end;
k = floor((n+1)/2);
p0 = p(1:k);
p1 = p((k+1):(n+1));
q0 = q(1:k);
q1 = q((k+1):(n+1));
r0 = karatsuba(p0,q0);
r2 = karatsuba(p1,q1);
r1 = karatsuba(add(p0,p1),add(q0,q1));
r1 = add(r1,-r0); r1 = add(r1,-r2);
r = add( r0, [zeros(k,1);r1] );
r = add( r, [zeros(2*k,1);r2] );
```

7.3 The Matlab code for defining elliptic curve ecc.m

Below is the matlab code for the ecc.m which performed elliptic curve addition over the real numbers.
Let E be the elliptic curve $y^2 = x^3 + Ax + B$ and let
P1 = (x1, y1),P2 = (x2, y2). The m-file will then produce
P1 + P2 = P3 = (x3, y3)
where + is the elliptic curve addition operation over E. The user must input the coordinates x1, y1, x2, y2 and, if P1 = P2, also the parameter A.

```
%---------defining elliptic curve
p = 23;
aconst = 1;
bconst = 1;
a = 1;
b = 1;
z =  mod(4*(a^3) + 27*(b^2),p);
%-------- determining quadratic residues
z23 = [1:1:p-1]; %reduced set of residues
x = [1:1:((p-1)/2)];
for x = 1 : 1 : (p-1)/2
   q23_1(x) = mod(x^2,p);
end
for x = 1 : 1 : (p-1)/2
   q23_2(x) = mod((p-x)^2,p);
end
q23 = intersect(q23_1,q23_2);%quadratic residues
%Defining elliptic curve points
for i = 0 : 1 : p-1
   y2(i+1) = mod((i^3+i+1),p);

end
%Elliptic group
y_1 = [];
y_2 = [];
for i = 1 : length(y2);
   [v,idx] = find(y2(i)==q23_1);
   if(isempty(idx)==0)
      y_1 = [y_1 idx];
      y_2 = [y_2 p-idx];
   else
     y_1 = [y_1 NaN];
      y_2 = [y_2 NaN];
   end
end
e = [];
j = 0;
for i = 1 : p
   if(y_1(i)>0)
      j = j+1;
      a = [i-1 y_1(i)];
      b = [i-1 y_2(i)];
      e(j,:) = [a b];
   end
end
x = round(1 + (size(e,1) - 1) * rand);
y = gencolnum;
xi = e(x,y);
yi = e(x,y+1);
ei = [xi yi];     %required elliptical curve point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

7.5This matlab code is for point multiplication eccmulti.m

```
function EJ = pointmulti(P,G,a)
n = floor(G/2);
r = 2^n;
while(r>G)
   n = n - 1;
   r = 2^n;
end
ad = G - r;

if(ad > 2)
   n1 = floor(ad/2);
   r1 = 2^n1;
   while(r1>ad)
      n1 = n1 - 1;
      r1 = 2^n1;
   end
   ad1 = ad - r1;
   PP = P;
   for i = 1 : n1
      S = PP(1) + ( PP(2)/PP(1) );
      XL = (S^2) + S + a;
      K = karatsuba((S+1),XL);
      YL = PP(1)^2 + K;
      PP(1) = XL;
      PP(2) = YL;
   end
   if(ad1==1)
      S = ( PP(2) +  P(2) ) / ( PP(1) + P(1) );
      tmpEJ(1) = S^2 + S + PP(1) + P(1) + a;
      tmpEJ(2) = ( PP(1) + P(1) ) + ( tmpEJ(1) + PP(2) );
   end
end

PP = P;
for i = 1 : n
   S = PP(1) + ( PP(2)/PP(1) );
   XL = (S^2) + S + a;
   K = karatsuba((S+1),XL);
   YL = PP(1)^2 + K;
   PP(1) = XL;
   PP(2) = YL;
```

```
end
if(ad==1)
   S = ( PP(2) +  P(2) ) / ( PP(1) + P(1) );
   EJ(1) = S^2 + S + PP(1) + P(1) + a;
   EJ(2) = ( PP(1) + P(1) ) + ( EJ(1) + PP(2) );
elseif(ad==2)
   S = P(1) + ( P(2)/P(1) );
   XL = (S^2) + S + a;
   K = karatsuba((S+1),XL);
   YL = P(1)^2 + K;
   PPP(1) = XL;
   PPP(2) = YL;
   S = ( PP(2) +  PPP(2) ) / ( PP(1) + PPP(1) );
   EJ(1) = S^2 + S + PP(1) + PPP(1) + a;
   EJ(2) = ( PP(1) + PPP(1) ) + ( EJ(1) + PP(2) );
elseif(ad==0)
    EJ(1) = PP(1);
    EJ(2) = PP(2);
else
   S = ( PP(2) +  tmpEJ(2) ) / ( PP(1) + tmpEJ(1) );
   EJ(1) = S^2 + S + PP(1) + tmpEJ(1) + a;
   EJ(2) = ( PP(1) + tmpEJ(1) ) + ( EJ(1) + PP(2) );
end
```

7.4 This matlab code is for encryption and decryption

encrydecryp.m

```
g = round(1 + (10 -1) * rand);%private key
ej = pointmulti(ei,g,aconst);

load ASCIItable


ptest = 'welcome'; %input('Enter any string :')


temph  = [];
for i = 1 : length(ptest)
   for j = 1 : size(ASCIItable,1)
      if(ptest(1,i) == ASCIItable{j,1})
         temph = [temph ASCIItable{j,2}];
      end
   end
end
tempi = round(1 + (length(temph) - 1) * rand);
h = temph(tempi);
while(h==0)
   tempi = round(1 + (length(temph) - 1) * rand);
   h = temph(tempi);
end
ci = pointmulti(ei,h,aconst);
cjpart = pointmulti(ej,h,aconst);
temp  = [];
for i = 1 : length(ptest)
   for j = 1 : size(ASCIItable,1)
      if(ptest(1,i) == ASCIItable{j,1})
         temp = [temp ; ASCIItable{j,2}];
      end
   end
end
temp

for i = 1 : length(ptest)
   cji(i,:) = pointadd(temp(i,:),cjpart,aconst);
end
%Decryption
for i = 1 : size(cji,1)
   desc(i,:) = pointsub(cji,pointmulti(ci,g,aconst),aconst);
end
desc
```

8. CONCLUSION

We provide a brief overview of elliptic curve cryptography and obtain the ASCII table with Karatsuba Multiplier for fast encryption and decryption. The strength of encryption depends on its key, if we use the alphabetical table then there will be no impact on strength and runtime performance. Runtime will be faster by this process, i.e use of ASCII table will provide better performance .This process and its implementation have been developed by ourselves .Further we are developing Elliptic curve cryptography by ASCII codes with some fast algorithms and their implementation.

9.REFERENCES

[1] Koblitz N., Menezes A.J., and Vanstone S.A. The state of elliptic curve cryptography. *Design, Codes, and Cryptography.* Vol 19, Issue 2-3, 2000, page 173-193.

[2] A. Karatsuba and Y. Ofman, Multiplication of multidigitnumbers on automata, *Sov. Phys. Dokl.*, Vol.7, 1963, 595-596.

[3] Nedjah, N. and Mourelle, L.M. (Eds.), Embedded Cryptographic Hardware:Methodologies andApplications, Nova Science Publishers, Hauppauge,NY, USA, 2004.

[4].S. Bajracharya, **C. Shu**, K. Gaj, and T. El-Ghazawi, Implementation of Elliptic Curve Cryptosystems over GF(2n) in Optimal Normal Basis on a Reconfigurable Computer.

[5]. Menezes A.J., Teske E., and Weng A. *Weak fields for ECC.*CORR 2003-15, Technical Report, University of Waterloo, 2003..

[6]. Rivest R., Schamir A. and Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21, 1978, 120-126.

[7]. C. Shu, S. Kwon, and K. Gaj, Reconfigurable Computing Approach for Tate Pairing Cryptosystems over Binary Fields *submitted to IEEE Transactions on Computers.*

[8]. Certicom. Information on the Certicom ECCchallenge, http://www.certicom.com/research/ecc hallenge.html

[9] ElGamal, T., A public key cryptosystem and a signature scheme based on discrete logarithm,IEEE Trans. Inform, Theory, IT-31, no.4, pp469-472, July 1985.

[10]T.N.Shankar,G.Sahoo,Cryptography with ellipitic curves ,International Journal of Computer Science And Applications ,page 38-42,Vol.2 No.1.

[11]T.N.Shankar,G.Sahoo,Cryptography with ASCII codes ,International Journal of Secure Digital Age Information Vol.1 No.2.

[12] Computer organization by Moris Mano,PHI publications.