Implementation of Elliptic Curve Digital Signature Algorithm Using Variable Text Based Message Encryption

[1]Jayabhaskar Muthukuru' [2]Prof. Bachala [3]Sathyanarayana
[1,2] Research Scholar, Department of Computer Science & Technology,
[1,2],Professor, Sri Krishnadevaraya University, INDIA

Abstract:
Digital Signatures are considered as digital counterparts to handwritten signatures, and they are the basis for validating the authenticity of a connection. It is well known that with the help of digital signature, forgery of digital information can be identified and it is widely used in e-commerce and banking applications. Elliptic curve digital signatures (ECDSA) are stronger and ideal for constrained environments like smart cards due to smaller bit size, thereby reducing processing overhead. We have implemented ECDSA over Elliptic Curve (EC) P-192 and P-256 using various Text Message encryptions which are Variable Size Text Message(VTM), Fixed Size Text Message(FTM) and Text Based Message(TBM) encryption methods and compared their performance.

Keywords: Digital Signature, Elliptic Curve Digital Signature Algorithm, Elliptic Curve Cryptography, ECDLP.

1. Introduction

Cryptography is the branch of cryptology dealing with the design of algorithms for encryption and decryption, intended to ensure the secrecy and/or authenticity of message. The Digital Signature Algorithm (DSA) was proposed in August 1991 by the U.S. National Institute of Standards and Technology (NIST). Digital signature authentication schemes provide secure communication with minimum computational cost for real time applications, such as electronic commerce, electronic voting, etc. The sender generates the signature of a given message using his secret key; the receiver then verifies the signature by using sender's public key. The ECDSA have a smaller key size, which leads to faster computation time and reduction in processing power, storage space and bandwidth. This makes the ECDSA ideal for constrained devices such as pagers, cellular phones and smart cards. The Elliptic-Curve Digital Signature Algorithm (ECDSA) is a Digital Signature Scheme based on ECC. ECDSA was first proposed in 1992 by Scott Vanstone in response of NIST (Nation Institute of Standards and Technology) request for public comments on their proposal for Digital Signature Schemes[1].

Digital Signature authenticated schemes, have the following properties.

1. **Confidentiality.** Secret information shared between sender and receiver; any outsider cannot read the information.
2. **Authentication.** The sender imprints his identity by means of the digital signature, which only the designated receiver can unravel and verify. An anonymous adversary cannot send a malicious message impersonating the genuine sender, because he does not have the necessary tools to generate the signature.
3. **Non-repudiation.** The signature firmly establishes the identity of the sender. The sender cannot deny having sent the message and the signature.

In this paper we discuss ECC in detail and ECDSA Implementation with different Text Message encryption methods and compared the results.

2. Elliptic Curve Discrete Logarithm Problem

An elliptic curve E, [2] defined over a field K of characteristic $\neq 2$ or 3 is the set of solutions $(x, y) \in K'$ to the equation

$$y^2 = x^3 + ax + b \quad (1)$$

$a, b \in K$ (where the cubic on the right has no multiple roots).

Two nonnegative integers, a and b, less than p that satisfy:

$$4a^3 + 27b^2 \pmod p = 0 \quad (2)$$

Then $E_p(a, b)$ denotes the elliptic group mod p whose elements (x, y) are pairs of nonnegative integers less than p satisfying:

$$y^2 = x^3 + ax + b \pmod p \quad (3)$$

together with the point at infinity O.

The elliptic curve discrete logarithm problem(ECDLP) can be stated as follows. Fix a prime p and an elliptic curve.

$$Q = xP \quad (4)$$

where xP represents the point P on elliptic curve added to itself x times. Then the elliptic curve discrete logarithm problem is to determine x given P and Q. It is relatively easy to calculate Q given x and P, but it is very hard to determine x given Q and P.

ECC is based on ECDLP. ECDH and ECDSA are cryptographic schemes based on ECC. The best known algorithm for solving ECDLP is Pollard-Rho algorithm which is fully exponential having a running time of $\sqrt{(\Pi * n/2)}$.

3. Elliptic Curve Cryptography

The Elliptic curve cryptosystems (ECC) were invented by Neal Koblitz [2] and Victor Miller[3] in 1985. They can be viewed as elliptic curve analogues of the older discrete logarithm (DL) cryptosystems in which the subgroup of Z_p^* is replaced by the group of points on an elliptic curve over a finite field. The mathematical basis for the security of elliptic curve cryptosystems is the computational intractability of the elliptic curve discrete logarithm problem (ECDLP) [4].

ECC is a relative of discrete logarithm cryptography. An elliptic curve E over Z_p as in Figure 1 is defined in the Cartesian coordinate system by an equation of the form:

$$y^2 = x^3 + ax + b \quad (5)$$

where $a, b \in Z_p$, and $4a^3 + 27b^2 \pmod p \neq 0 \pmod p$, together with a special point O, called the point at infinity. The set $E(Z_p)$ consists of all points (x, y), $x \in Z_p$, $y \in Z_p$, which satisfy the defining equation, together with O.

Each value of a and b gives a different elliptic curve. The public key is a point on the curve and the private key is a random number. The public key is obtained by multiplying the private key with a generator point G in the curve.

The definition of groups and finite fields, which are fundamental for the construction of elliptic curve cryptosystem are discussed in next subsections.

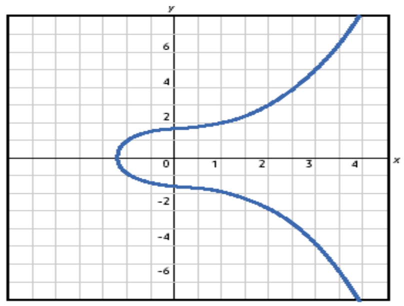

Figure 1. An Elliptic Curve

3.1. Groups

A group with an operation * is defined on pairs of elements of G. The operations satisfy the following properties:

- Closure: $a * b \in G$ for all $a, b \in G$
- Associativity: $a * (b * c) = (a * b) * c$ for all $a, b, c \in G$
- Existence of Identity: There exists an element $e \in G$, called the identity, such that $e * a = a * e = a$ for all $a \in G$.
- Existence of Inverse: For each $a \in G$ there is an element $b \in G$ such that $a * b = b * a = e$. The element b is called the inverse of a.

Moreover, a group G is said to be abelian if $a * b = b * a$ for all $a, b \in G$. The order of a group G is the number of elements in G.

3.2. Finite Field

A finite field consists of a finite set of elements together with two binary operations called addition and multiplication, which satisfy certain arithmetic properties. The order of a finite field is the number of elements in the field. There exists a finite field of order q if and only if q is a prime power. If q is a prime power, then there is essentially only one finite field of order q; this field is denoted by F_q. There are, however, many ways of representing the elements of F_q. Some representations may lead to more efficient implementations of the field arithmetic in hardware or in software. If $q = p^m$ where p is a prime and m is a positive integer, then p is called the characteristic of F_q and m is called the extension degree of F_q.

3.2.1. Prime Field F_p

Let p be a prime number. The finite field F_p called a prime field, is comprised of the set of integers {0,1,2,….,p-1} with the following arithmetic operations:

- Addition: If $a, b \in F_p$ then $a + b = r$, where r is the remainder when a + b is divided by p and $0 \le r \le p-1$ known as addition modulo p.
- Multiplication: If $a, b \in F_p$ then $a.b = s$, where s is the remainder when a.b is divided by p and $0 \le s \le p-1$ known as multiplication modulo p.

- Inversion: If a is non-zero element in F_p, the inverse of modulo a modulo p, denoted by a^{-1}, is the unique integer $c \in F_p$ for which a.c = 1.

3.2.2. Binary Field $F_2{}^m$

The field $F_2{}^m$, called a characteristic two finite field or a binary finite field, can be viewed as a vector space of dimension m over the field F_2 which consists of the two elements 0 and1. That is, there exist m elements $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ in $F_2{}^m$ such that each element α can be uniquely written in the form:

$$\alpha = a_0\,\alpha_0 + a_1\,\alpha_1 + \ldots\ldots\ldots + a_{m-1}\,\alpha_{m-1}, \text{ where } a_i \in \{0,1\}$$

Such a set $\{\alpha_0, \alpha_1, \ldots, \alpha_{m-1}\}$ is called a basis of $F_2{}^m$ over F_2. Given such a basis, a field element α can be represented as the bit string ($a_0 + a_1 \ldots\ldots\ldots + a_{m-1}$) Addition of field elements is performed by bitwise XOR-ing the vector representations. The multiplication rule depends on the basis selected. ANSI X9.62 permits two kinds of bases: polynomial bases and normal bases.

3.2.3. Domain Parameters

The domain parameters for ECDSA consist of a suitably chosen elliptic curve E defined over a finite field F_q of characteristic p, and a base point $G \in E(Fq)$. Domain parameters may either be shared by a group of entities, or specific to a single user. To summarize, domain parameters are comprised of:

1. A field size q, where either q = p, an odd prime, or $q = 2^m$
2. An indication FR (field representation) of the representation used for the elements of F_q
3. (optional) a bit string seed E of length at least 160 bits
4. Two field elements a and b in F_q which define the equation of the elliptic curve E over F_q' (i.e., $y^2 = x^3 + ax + b$ in the case p > 3, and $y^2 + xy = x^3 + ax + b$ in the case p = 2)
5. Two field elements x_G and y_G in F_q which define a finite point $G = (x_G, y_G)$ of prime order in $E(F_q)$
6. The order of the point G, with $n > 2^{160}$ and $n > 4\sqrt{q}$ and
7. The cofactor $h = \#E(F_q)/n$

3.3. Elliptic Curve Operations over Finite Fields[8]

The main operation is Point multiplication is achieved by two basic elliptic curve operations.

i. Point addition, adding two points P and Q to obtain another point R i.e. R= P + Q.
ii. Point doubling, adding a point P to itself to obtain another point R i.e. R = 2P.

3.3.1. Point Addition

Point addition is the addition of two points P and Q on an elliptic curve to obtain another point R on the same elliptic curve. Consider two points P and Q on an elliptic curve as shown in Figure 2. If $P \neq -Q$ then a line drawn through the points P and Q will intersect the elliptic curve at exactly one more point –R. The reflection of the point –R with respect to x-axis gives the point R, which is the result of addition of points P and Q. Thus on an elliptic curve R = P + Q. If Q = -P the line through this point intersect at a point at infinity O. Hence P + (-P) = O. A negative of a point is the reflection of that point with respect to x-axis.

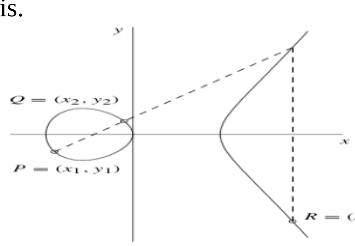

Figure 2: Point Addition

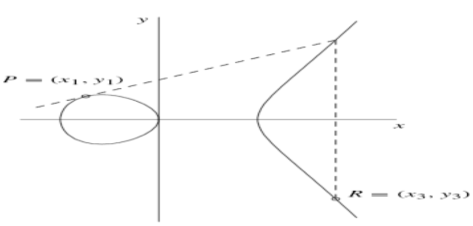

Figure 3: Point Doubling

3.3.2. Point Doubling

Point doubling is the addition of a point P on the elliptic curve to itself to obtain another point R on the same elliptic curve To double a point J to get L, i.e. to find R = 2P, consider a point P on an elliptic curve as shown in Figure 3. If y coordinate of the point P is not zero then the tangent line at P will intersect the elliptic curve at exactly one more point –R. The reflection of the point –R with respect to x-axis gives the point R, which is the result of doubling the point P, i.e., R = 2P. If y coordinate of the point P is zero then the tangent at this point intersects at a point at infinity O. Hence 2P = O when $y_j = 0$. Figure 3 shows point doubling.

3.3.3. Algebraic Formulae over F_p

Let p be a prime in F_P and a, b$\in$ FP such that $4a^3 + 27b^2 \neq 0$ mod p in F_P, then an elliptic curve E (F_P) is defined as

$$E(F_P) := \{ p(x, y), x, y \in F_P \}$$

Such that $y^2 = x^3 + ax + b$ mod p together with a point O, called the point at infinity. Below is the definition of addition of points P and Q on the elliptic curve E (F_P). Let P(x_1, y_1) and Q(x_2, y_2) then

$$R = P+Q = \begin{cases} & \text{If } x_1 = x_2 \text{ and } y_2 = -y_1 \\ Q = Q+P & \text{If } P = O \\ (x_3, y_3) & \text{otherwise} \end{cases}$$

$$\text{Where } x_3 = \begin{cases} \lambda^2 - x_1 - x_2 & \text{If } P \neq \pm Q \text{ (Point Addition)} \\ \lambda^2 - 2x_1 & \text{If } P = Q \text{ (Point Doubling)} \end{cases}$$

$$y_3 = \lambda(x_1 - x_3) - y_1, \text{ and}$$

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{If } P \neq \pm Q \text{ (Point Addition)} \\ \dfrac{3x_1^2 + a}{2y_1} & \text{If } P = Q \text{ (Point Doubling)} \end{cases}$$

The point p(x, -y) is said to be the negation of p(x, y).

3.3.4. Algebraic Formulae over F_2^m

Denote the (non-super singular) elliptic curve over F_2^m by E (F_2^m). If a, b $\in F_2^m$ such that b $\neq$ 0 then

$$E(F_2^m) = \{p(x, y), x, y \in F_2^m\}$$

such that $y^2 + xy = x^3 + ax^2 + b \in F_P^m$ together with a point O, called the point at infinity.
The addition of points on E (F_2^m) is given as follows: Let P(x_1, y_1) and Q(x_2, y_2) be points on the elliptic curve E(F_2^m), then

$$R = P+Q = \begin{cases} O & \text{If } x_1 = x_2 \text{ and } y_2 = -y_1 \\ Q = Q+P & \text{If } P = O \\ (x_3, y_3) & \text{otherwise} \end{cases}$$

$$\text{Where } x_3 = \begin{cases} \lambda^2 + \lambda + x_2 + x_1 + a & \text{If } P \neq \pm Q \text{ (Point Addition)} \\ \lambda^2 + \lambda + a & \text{If } P = Q \text{ (Point Doubling)} \end{cases}$$

$y_3 = \lambda (x_1 + x_3) + x_3 + y_1$ and

$$\lambda = \begin{cases} \dfrac{y_2 + y_1}{x_2 + x_1} & \text{If } P \neq \pm Q \text{ (Point Addition)} \\ x_1 + \dfrac{x_1}{y_1} & \text{If } P = Q \text{ (Point Doubling)} \end{cases}$$

4. Implementation

This paper presents VTM Encryption, VTM decryption [5], ECDSA key generation, signature generation and signature verification algorithms [8] and ECDSA was implemented over Elliptic Curve (EC) P-192 and P-256 using Text Message Encryption methods which are VTM [5], FTM[5] and TBM [6] encryption methods and compared their performance.

Algorithm-1

VTM Encryption Algorithm[5]

NOTATION: TM - Text message

M - Message units
VS - variable size
IV - Initial Vector
k - Auxiliary base parameter
XRM - XORed message
Block – a word with followed space

INPUT: sextuple T = (p, a, b, G, n, h), Text Message

OUTPUT: Encrypted Message

Begin
  n = wordCount(TM)
  for i = 1 to n **do**
    XRM = IV $\oplus$ Block[i]
    M = ASCII(XRM)
    for j = 0 to k-1 **do**
      let $x_j = M * K + j \bmod p$
      if $z_j = x_j^3 + x_j + b$ has a square root mod p **then**
        break
      end if
    end for
    if j < k **then**
      compute y_j a square root of z_j mod p
      map M to (x_j, y_j)
    else
      output "unsuccessful in attempt to map M to an EC point"
    end if
    Cm[i] = { kG ,Pm + kP_B}
    IV = XRM
  end for
End

Algorithm-2

VTM Decryption Algorithm[5]

INPUT: sextuple T = (p, a, b, G, n, h), Encrypted Message

OUTPUT: Decrypted/Plain text Message

Begin
for i = 1 to n **do** //where n is number of cipher texts
Pm(x, y) = Pm + K(n_BG) - n_B(kG) // n_B receivers private key
M = x/k
Dm = Text(M) // M is decimal value of base 256 format
TM[i] = Dm ⊕ IV
IV = Dm
TM = TM || TM[i]
end for
End

Algorithm-3

ECDSA Key pair generation Algorithm[8]

INPUT: Domain parameters D= (q, FR, a, b, G, n, h).

OUTPUT: Public key Q, private key d.

Select d∈ [1, … , n-1]
Compute Q = dG
Return (Q, d)

Algorithm-4

ECDSA Signature Generation Algorithm[8]

INPUT: Domain parameters D= (q, FR, a, b, G, n, h) , private key d, Encrypted message m'.

OUTPUT: Signature (r,s)

begin
repeat
k = Random[1, … , n-1] // select random value
r = x-coord([k]G) mod n
e = H(m')
s = k^{-1}(e+dr) mod n
until r ≠ 0 and s ≠ 0
return (r,s).
end

Algorithm-5

ECDSA Signature Verification Algorithm[8]

INPUT: Domain parameters D= (q, FR, a, b, G, n, h) , public key Q, Encrypted Message m', Signature (r, s).
OUTPUT: Acceptance or rejection of the signature.
begin
if r, s ∉ [1,…, n] **then**
Return (“Reject the signature”)
end if
e = H(m')
w = s^{-1} mod n
u_1 = ew mod n
u_2 = rw mod n

x = $u_1G + u_2Q$
if x = ∞ **then**
Return ("Reject the signature")
end if
v = x-coord(X) mod n
if v = r **then**
Return ("Accept the signature")
else
Return ("Reject the signature")
end if
end.

Elliptic Curve based Signature Generation & Signature Verification processes are described below and the same is represented in graphical format in figure 4 and figure 5.

Signature Generation steps:

1. Encrypt the message using EC Encryption algorithm which is VTM/FTM/TBM
2. Compute signature for Encrypted message using Algarithm-4
3. Send the digitally signed message

Signature Verification Steps:

1. Verify Signature using Algarithm-5.
2. If verification fails then reject the signature
3. If verification success, then decrypt the message using respective EC Decryption Algorithm.

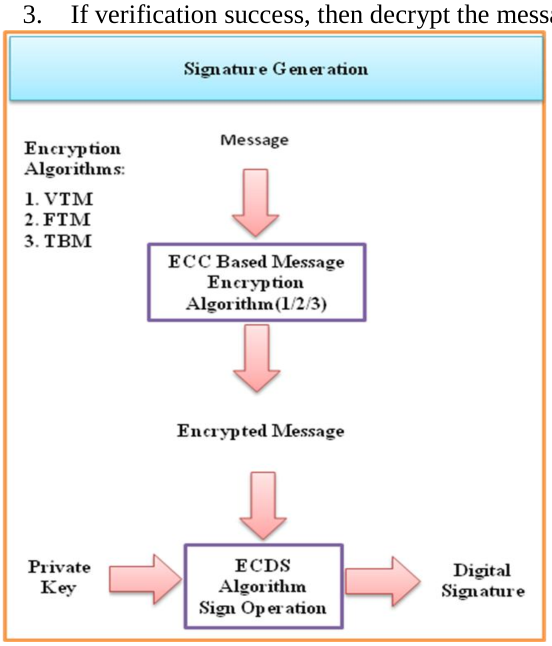

Figure 4: Signature Generation Process

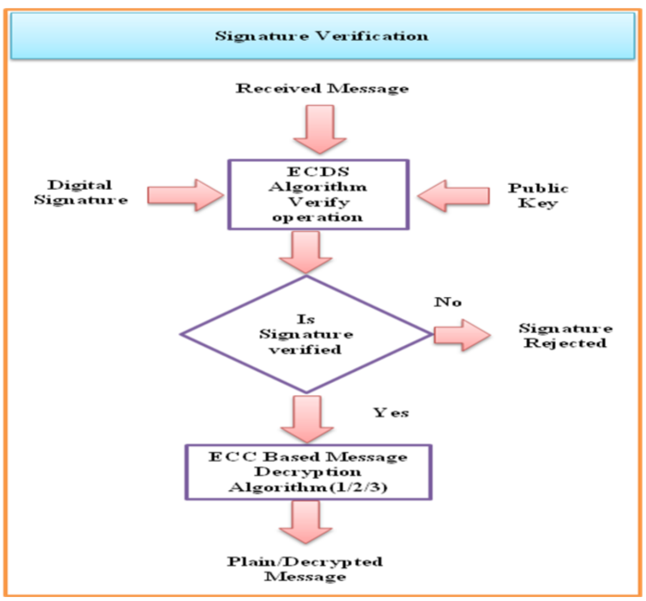

Figure 5: Signature Verification Process

5. Results and Discussion

In this section represents implementation results of ECDSA using VTM encryption over EC P-192 and P-256.

5.1. Results over Elliptic Curve P-192

Message m = "Test Run"
Private key = 2055107281
Public Key = (5841942716391479201550342297351085963270983519924994377602,
5584890377300947026793868981513336619407548239394095574193)
This message encrypted and follows Signature Generation and Verification as mentioned below.
Encrypted message hash value H(E(m)) = -2682108996977278156968408606235438945161064554

- ECDSA SIGNATURE as follows:

Select k= 1583021364
Compute kG = (3792194627815960440118002914594551166312864178888962630882,

28911906596206560599907180226621467285648536055401680019 82)

r = 37921946278159604401180029145945511663128641788889626308 82

Compute s = k^{-1} (e + dr) mod n = 3411184681610252308390502359065554562708605093739075483483

Signature for the message m is (r, s).

- ECDSA VERIFICATION as follows:

Compute w = 57774801458036697415734236889261769794170825052710323602 68

Compute u_1 = 46664225272490341000420229463370900085105972771841113036 96

u_2 = 44559079274298864732772044749902368531248771713356612716 49

u_1G = (39297089899694676971974867166721224469423156320948310433 67,
45370034565711033802845048137217920961191980475439594916 71)

u_2Q = (12776617158002053480674207660168064759541336269296963833 70,
43808084603875676491070542897325858868480882061254487424 47)

v = 37921946278159604401180029145945511663128641788889626308 82

We obtain v = r, that is accept the signature.

5.2. Results over Elliptic Curve P-256

Message m= "How are you?"

Private Key = 978425864

Public Key = (118910487909274429022743485742135581553673510998540082125096949934594470938 22,
136698797209684711142721957596171372481001364004993589753744001635050991639 86)

This message encrypted and follows Signature Generation and Verification as mentioned below.

Encrypted message hash value H(E(m)) = 53770309037964977040219539705106232306909249184 6

- ECDSA SIGNATURE as follows:

Select k= 115792089210356248762697446949407573529996955224135760342422259061068383502243

Compute

KG = (865008812241664832279252673133542372930184288124092450477788075098073585550 53,
395790536103464344705325064380117869670575066132236893145938518519821175997 76)

r = 865008812241664832279252673133542372930184288124092450477788075098073585550 53

Compute s = k^{-1} (e + dr) mod n

= 104389700715501732796614779737855463749375844486540618622018054702970561091708

Signature for the message m is (r, s).

- ECDSA VERIFICATION as follows:

Compute w = 106506396977556145535418054052339447393078832993181450002668470251312371474276

Compute u_1 =43824495211803284954034352427133274304161118431427286644319226927046995292 09

u_2=576926169823111609841763667288476477338005393627061470291328150661625922194 39

u_1G = (101474627893392564150949213703200203728873111984 8 92002825714765996844262058436,
609374231091592309903483369499808 0 456436196569064621167172651499915155479540 8)

u_2Q = (109322103145683055628956971282445177307378355734712278598030249871906512163766,
427536393825241362742313342843055722126028431868422360431368270793952995525 47)

v = 865008812241664832279252673133542372930184288124092450477788075098073585550 53

We obtain v = r, that is accept the signature.

In the same way we have used FTM and TBM encrypted message for Signature generation and signature verification. ECDSA using Variable Size Text Message Encryption is better in performance aspect when compare with the other two methods and the results comparison is presented graphically in the next section.

6. Comparison Of ECDSA Using Various Text Based Cryptosystems

We compare the results of ECDSA using Variable Size Text Message(VTM) Encryption[5] with ECDSA using Fixed Size Text Message(FTM) Encryption[5] and Text Based Message(TBM) Encryption[6]. Figure 6 and Figure 7 presents total time taken for Signature Generation and Signature Verification when we use different text based encryption methods in ECDSA implementation. From Figure 6 and Figure 7, performance of ECDSA using Variable Size Text Message Encryption is better when compare with ECDSA using FTM Encryption and TBM Encryption. The reason is VTM based ECDSA used less number of point additions and multiplications compare with other two methods. Performance of ECDSA is inversely proportional to key size, and security of the system depends on key size.

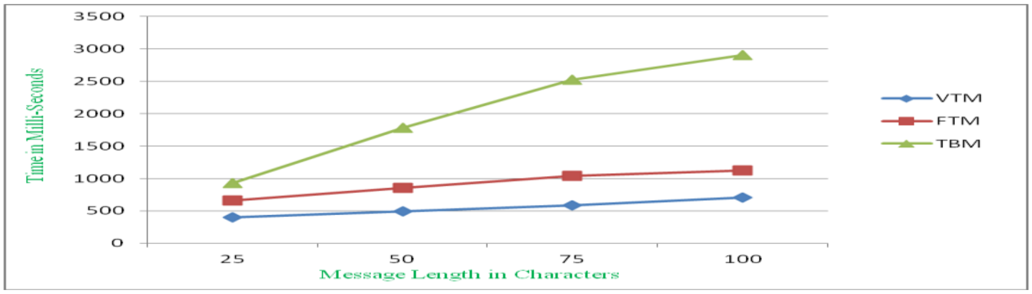

Figure 6: Performance comparison of various ECDSA methods for over EC P-192

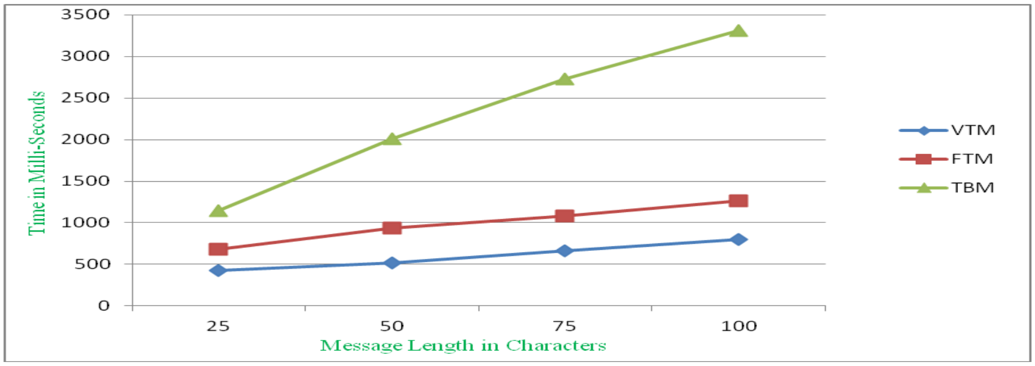

Figure 7: Performance comparision of various ECDSA methods for over EC P-256

7. Conclusion

In this paper we have implemented ECDSA for various domain parameters, after observing the results when the key size increases then complexity increases and performance decreased. After comparing VTM, FTM and TBM based ECDSA methods, ECDSA using Variable Text Message Encryption is better when comparing with Fixed Length Text Message and Text Based Encryption used ECDSA. The main reason is, the speed of scalar multiplication which plays an important role in the efficiency of whole system [7]. In VTM based ECDSA method, number of scalar multiplications are reduced, so this method is efficient when compared with FTM and TBM based methods.

References

[1] Navneet Randhawa, Lolita Singh, A Systematic Way to Provide Security for Digital Signature Using Elliptic Curve Cryptography, IJCST Vol.2, Issue 3, Sep-2011, 185-188
[2] Koblitz, N., 1987. Elliptic curve cryptosystems. Mathematics of Computation 48, 203-209.
[3] Miller, V., 1985. Use of elliptic curves in cryptography. CRYPTO 85.
[4] Certicom ECC Challenge. 2009. Certicom Research
[5] Jayabhaskar Muthukuru, Bachala Sathyanarayana, Fixed and Variable Size Text Based Message Mapping Techniques Using ECC, GJCST Vol.12, Issue 3, Feb-2012, 25-30.
[6] S. Maria Celestin Vigila , K. Muneeswaran "Implementation of Text based Cryptosystem using Elliptic Curve Cryptography", IEEE Sep-2009, pp. 82-85.
[7] Harsandeep Brar , Rajpreet Kaur, "Design and Implementation of Block Method for Computing NAF" IJCA, Volume 20– No.1, April 2011, pp. 37-41.
[8] Hankerson, D., Menezes, A., Vanstone, S., Guide to Elliptic Curve Cryptography (Springer, 2004).