DOI: http://dx.doi.org/10.26483/ijarcs.v9i3.6104

ISSN No. 0976-5697

Volume 9, No. 3, May-June 2018

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

TIME COMPLEXITY ANALYSIS OF RSA AND ECC BASED SECURITY ALGORITHMS IN CLOUD DATA

D.Pharkkavi and Dr. D. Maruthanayagam
[1]Research Scholar, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India
[2] Head/Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India
dr.d.maruthanayagam@gmail.com

Abstract: Cloud computing is being heralded as an important trend in information technology throughout the world. Data security has a major issue in cloud computing environment; it becomes a serious problem due to the data which is stored diversely over the cloud. Data privacy and security are the two main aspects of user's concern in cloud information technology. Now-a-days, the cloud data security method uses the symmetric encryption and asymmetric encryption algorithms with their strong authentication techniques. In this paper, we discuss and compare the performances for number of existing security techniques used to provide security in the field of cloud computing on the basis of different parameters. It will be useful to enhance the security of data storage in a cloud environment and also to find proposed a novel security algorithm.

Keywords: Cloud Computing, Security, Cryptography, ECC, RSA, ECDH and ECDSA

I.INTRODUCTION

Cloud computing utilizes with an attractive tag line 'pay-as-you-use' for attracting users to its great elasticity and scalability of resources at relatively low cost. Evaluated to the construction of their own infrastructures, customers are capable to reduce on important expenditure procedure by securing storage, migrating computation and hosting onto the cloud. Even though this affords savings in terms of manpower and finance, it takes lots of new risks and challenges.

The authority of the cloud computing is considered with respect to its technological transformations and business benefits, the future enterprise applications are completely dependent on it. It has its individual benefits; however it has several risks and challenges like, Privacy Issues, Data Theft, Infected Application, Data Integrity, Data Location, Data Loss, Security on User Level, and Security on Vendor Level [1]. Nowadays cryptography is more useful than encryption and decryption. Authentication is a basic part of our daily life as the privacy protection. We use authentication throughout to process day-to-day lives when we sign our name to some document, where our agreements and decisions are communicated electronically for providing authentication.

A digital signature attaches a document to the processor using a particular key, while a digital timestamp connects a document to a particular time. The risk may be difficult to find its solution needs some secret knowledge like signing few digital documents or decrypting an encrypted message. Cloud uses various cryptographic techniques necessary for cloud security. A key is utilized for data encryption and data decryption. This supports in securely protecting integrity and confidentiality of data. It ensures to protect the security of data to be shared in cloud and allows data to be stored securely [2].

Many cryptographic algorithms are considered with two major categories.
a) Symmetric algorithms like DES, Triple DES, AES,
b) Asymmetric or public-key encryption algorithms like Diffie-Hellman, RSA, ECDH, ECC, ECDSA etc.

In symmetric key encryption, the sender who is transmitting the data and the receiver who is receiving the data to be share a key which is kept secret [3]. This is the way used to encrypt and decrypt the messages. In asymmetric key encryption, two keys are involved wherein one key is used for encryption (publicly available) and the other key is used for decryption (kept secret).

- **Attribute based encryption:** The secret key of a user and the cipher text are depending upon attributes by using the public-key encryption, (e.g. the kind of subscription he has, or the country he lives,). A user can encrypt a message under a policy and a public key. Decryption process will only handle work if the attributes related with the decryption key match the policy used to encrypt the message.
- **Cloud-managed-key:** An additional possible threat with conventional cryptographic techniques can be allowing users manage their decryption keys themselves. Additionally, if a user has not provided permissions as long to access data, after that it can decrypt data if he has the key. The cloud is managed the key using a cryptographic technique as a possible solution for this issue can be resolved
- **Identity based encryption:** The Identity-based encryption (IBE) is a kind of public-key encryption where in the public key of a user is some unique information about the identity of the user (e.g. a client electronic mail address). An ASCII string consider as a known identity value for any party that allows a public key generation. The corresponding private key's utilized by performing private key generator and also known as third party access. This type of encryption process is able

to cut down the complexity for utilizing both administrators and users

II.SECURITY ALGORITHMS

2.1 RSA

In RSA schema, integer performs with in the interval [0, n-1] such as block cipher, original message and cipher message. In which the encrypted message and original message are represented h*h square matrices in another schema. In this technique, they don't have any restriction for encryption and decryption order and also consider as more dynamic, efficient and scalable [4]. For the above security purpose, the hardware implementation of RSA schema utilizing the modular exponentiation [5] and also provide security and help to save to computation time and processing time. Due to the increasing demand of security issues in communication channel its essential to improve a new technological development and efficient hardware security module.

It is an encryption-decryption technique. It consists of plaintext and cipher text in the form of integers between 0 to n-1. This plain text is encrypted in blocks; each and every block has a binary value which should be less than n.

This algorithm is done in three steps:

- Key generation
- Encryption
- Decryption

Key Generation:

In key generation consider two prime numbers (i.e.) p and q. it consists of public key and a private key. The public key will be known to everyone. Calculate the value of n. select a random encryption key e calculates the gcd and it should be equal to 1. Then find the decryption key d. finally calculate the public key and private key. The plain text is encrypted in blocks, with each block having a binary value less than some number n i.e., for block size i bits, $2^i<n<2^{i+1}$.

- Input: None
- Computations: Select two relatively prime numbers p and q.Where n=p*q and v-(p-1)*(q-1).
- Compute the integer d such that (d*e)%v=1.
- e is the integer.
- Output: n, e and d

Encryption process:

In the encryption process represent a plaintext in series of numbers modulo n. the encryption process to obtain cipher text C from plaintext M is very simple. It is formulated as: $C=M^e \bmod n$

Where C = cipher text

M = message text

E = public key

D = private key

The file will be encrypted by sending a symmetric File Encrypted Key **(FEK)** simultaneously asymmetric public key will generated both will be combined and forms an encrypted FEK with a header file.

- Input: Integers n, e, M
- M is integer representation of the plain text.
- Computation: Let C be the integer representation of the cipher text. $C=(M^e \bmod n)$
- Output: Encrypted text or cipher text C.

Decryption process:

The reverse process of encryption will be decryption. It can be generated using the formula: $m= e^d \bmod n$.

Where C =cipher text

M=message text

E =public key

D =private key

- Input : d, n, C
- C is the cipher text.
- Computation: Let D be the decrypted text such that $D=(C^d \text{ Mod } n)$
- Output: D is the decrypted message.
- Public Key: {e, n}
- Private Key: {d, n}

Example:

i) Prime Number P = 211, Q = 233.

ii) RSA Modules N= 211 x 233 = 49163

$$\phi(n) = (211-1).(233-1) = 48720.$$

iii) Public key e = 2^16+1= 65537.

iv) Private key $d \equiv e^{-1} \pmod{48720} \equiv 44723$.

v) Message M="INDIA"

vi) Encryption E**(M) ≡ M** ^{65537}**(mod 49163**).

Vii) Decryption D**(M) ≡ M**^{44723} **(mod 49163**).

Viii) Bob sends Alice the message **"INDIA"** as follows:

- The Input text will be separated into segments of Size 1 (the symbol '#' is used as separator).

  I # N # D # I # A = 01001001 # 01001110 # 01000100 # 01001001 # 01000001

ix) Encryption into ciphertext c[i] = m[i]^e (mod N)
1001001010000100 # 111001101111100 # 000001001101110 # 1001001010000100 # 000110011110100

x) Alice decrypts the message by computing, Decryption into plaintext m[i] = c[i]^d (mod N)
000010110100110 # 000110110000100 # 011001110011011 # 000010110100110 # 1010011111111010

2.2 ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

To generate cryptographic algorithms [6] the ECC cryptographic scheme uses the properties of elliptic curves.

In the 1980s Koblitz and Miller proposed using the group points on an elliptic curve defined over a finite field in discrete logarithmic cryptosystems. An elliptic curve is the solution set over a non-singular cubic polynomial equation with two unknowns over a field F. In short terms it is a discredited set of solutions to a curve that is in the form:

$$y^2 = x^3 + ax + b$$

These curves holds the property that if you draw a straight line that intersects the curve in two points, it will also intersect the curve in a third point that is either on the curve or the point of infinity (also referred to as the neutral element). Another important property of elliptic curves is that they are symmetric over the x-axis. That means that if you have a point P(x, y) then -P will be (x, -y). Using these properties one can define some interesting and useful arithmetic rules. We will now briefly explain how point addition over elliptic curves is done, as this is used for key generation. Suppose that you have a point A and a point B on an elliptic curve and you want to perform an addition of these two points. Then you draw a line from A through B. This line will intersect the curve in a third point. Take this third point and mirror it over the x-axis and that will be the result of the addition [7][8].

ALGORITHM FOR ECC

There has to be some information that is publicly known to all the users, thus making it the public key cryptography. The publicly known entities are:-

1. From the equation of the elliptic curve, we need to know:-
   - The values of the constants a and b.
   - The value of m, where elliptic curve is defined over GF(2m).
2. The group of the elliptic curve.
3. A base point B, i.e. any point on the curve E that belongs to the group taken as a base.

The algorithms for different parts of ECC are:-

Key Generation Algorithm

- Randomly select an integer Apriv. It acts as the private key for A.
- Then generate Apub such that Apub = Apriv * B, where Apub is the public key for A.
- Randomly select an integer Bpriv. It acts as the private key for B.
- Then generate Bpub such that Bpub = Bpriv * B, where Bpub is the public key for B.
- Finally, A generates key, Ka = Apriv * Bpub
- B generates key, Kb = Bpriv * Apub

Signature Generation Algorithm

- Calculation of message digest with a HASH function, preferable SHA-1, where e is the message digest, m is the message such that e = HASHfun(m)
- Generate a random integer r and between 1 and n-1.
- The first of the signature, sign1 is calculated from sign1 = x mod n where x is the product of B with rand i.e. x = xcod(rand * B) where xcod is a function to get the x co-ordinate.
- But if sign1 is 0, then redo the previous step.
- The second part of the signature, sign2 is calculated from the equation sign2 = rand -1(e + (Apriv*sign1)(mod n)
- But if sing2 is 0, then re-generate r and follow the procedure again.
- The signature generated is a pair (sign1, sign2).

Signature Validation Algorithm

- Check if sign1 and sign2 lie between the range of 1 and n-1. If not, the signature is not valid.
- Calculate the message digest from the received message with the same hash function, e = HASHfun(m).
- Calculate var1, where var1 = sign2 1(mod n)
- Calculate var2, such that var2 = (e*var1) mod n
- Calculate var3, such that var3 = (sign1*var1) mod n
- We then calculate X, such that X = (var2*B) + (var3*Apub)
- If sign1 (mod n) is equal to xcod(X), then signature is verified.

Encryption Algorithm

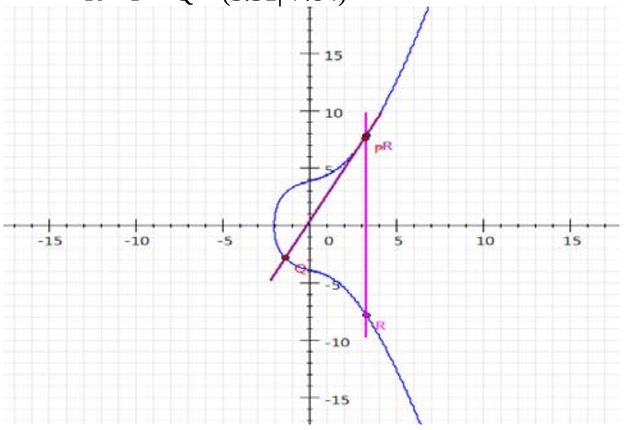
- The plain text M is mapped onto the elliptic curve at a point P.
- Generate a random integer rand between 1 and n-1.
- The cipher text is then encoded as a pair C, where C = [(rand * B),(P + (rand * Bpub)]

Decryption Algorithm

- Get x, where x = xcod(C).
- Calculate prod, where prod = Bpriv * x
- Calculate (P + (rand * Bpub)) prod), this gives the mapped point P
- Then un-map P to the plain text M

EXAMPLE:

1. Curve Size: Small , Curve Type: Real number, Curve attributes: a=3, b=15, Curve: $y^2 = x^3 + 3x + 15$,Point P = (3.2|7.57),Point Q = (-1.4|-2.83),Point R = P + Q = (3.31|-7.84)



2. Curve Size: Large, Curve Type: F(p), Select curve attributes: ANSI X9.62,Curve: prime192v1, Radix: 16 hexadecimal, Curve attributes: $y^2 = x^3 + 3x + 15$, where
a = fffffffffffffffffffffffffffffffefffffffffffffffc,
b = 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
p = fffffffffffffffffffffffffffffffeffffffffffffffff
Base point G: Point P
x = f505a38eb66ad677495e0713a37c4bd6fc826ee18c2de9d2
y= 421090022aaafa8976e9df198a61d84d384feb00dfe0c191

Base point G: point Q
x = e16e7fb72431ecb46f90235c19f8e8b499833be5fc7d3a49
y=4dae3baeed3af299d203621933943ee3f65f2e4139e5fe72
Point R : R = P + Q
x= f5c1bc8d5fd1ad01ba71edf255083f9185154bd5d644961c
y= f1247ab9e7171aeaf709657d87c9d368535c318201b0ffc8

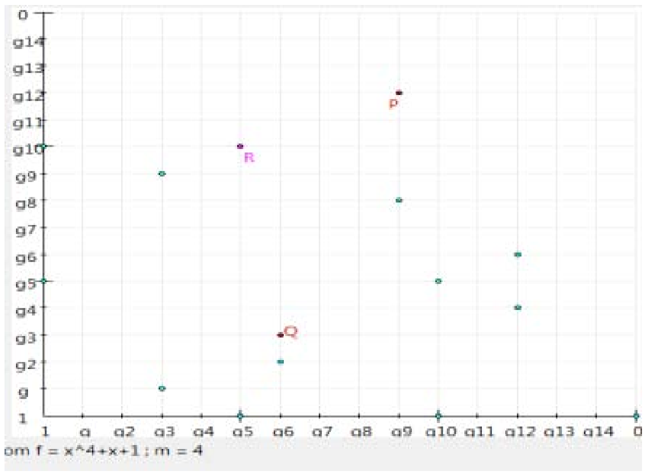


Figure 1: Time Complexity (MS) Comparison of RSA, ECC, ECDH and ECDSA

3. Curve Size: Large, Curve Type: F(2^m), Select curve attributes : ANSI X9.62, Curve: c2pnb163v1, Radix : 16 hexadecimal
a = 00000007 2546b543 5234a422 e0789675 f432c894 35de5242

b = 00000000 c9517d06 d5240d3c ff38c74b 20b6cd4d 6f9dd4d9
m = 163

Base Point P:
x = 00000007 af699895 46103d79 329fcc3d 74880f33 bbe803cb
y= 00000001 ec23211b 5966adea 1d3f87f7 ea5848ae f0b7ca9f

Base point Q:
X=00000007 3fbb1a82 e9f24136 3428217c 11d41746 30127d94
Y= 00000001 360090a6 d6df3487 da3808cf c571908e 60d8eb90
Point R : R = P + Q
X= 00000007 ee35173a 4ae9f401 c42fe4f6 01338998 bb745a37
Y= 00000003 6639922b a4e6c208 dc1f73b5 b137fc51 4a275c7d
4. Curve Size: Small, Curve Type: F(2^m), curve attributes : m=4, f = x^4+x+1,a=1,b=1,Curve: $y^2 + xy = x^3 + x^2 + 1$, Point P = (g9|g12), Point Q = (g6|g3), Point R = P + Q = (g5|g10)

2.3 ECDH – Elliptic Curve Diffie Hellman

Elliptic Curve Diffie Hellman (ECDH) represents an Elliptic Curve variant of using the standard Diffie Hellman algorithm. ECDH performs with a key agreement [9] [10]. It enables two parties to establish between the public key and the private key to exchange the shared key. Afterward, by using the shared key considers a key or the derived key and performs to encrypt subsequent communications using a symmetric-key cipher. For authentication purpose, each key pair of one of the party is trusted by using other party so as to provide secure authentication. Therefore, the systematic efforts are seem to be performed for providing a very faster public key cryptosystem and concurrently this scheme should be a very practical and protective, for the most constrained environments.

For example, a shared secret key is exchanged between E and F by using EC - Diffie hellman, both of EC domain parameters to agree up or to be obtained. Both sender and receiver contain a key pair which consists of a private key and a public key. A private key is randomly selected integer less then n, where n is the order of the curve and another public key is randomly selected as Q= d*G (G is the generator point). Let (d_E, Q_E) be the private-public key pair of E and (d_F, Q_F) be the private-public key of F.

- ✓ E Computes $K_E= (X_E, Y_E) = d_E * Q_F$
- ✓ F Computes $K_F= (X_F, Y_F) = d_F * Q_E$
- ✓ Given that $d_E * Q_F = d_E d_F G= d_F d_E G = d_F * Q_E$.
  Hence $K_E= K_F$ and hence $X_E = X_F$
- ✓ (Where G represents generator point)
- ✓ Thus the shared secret is K_E.

Diffie–Hellman key exchange system Using ECC

- Initially, Alex and Benny first select a finite field Fp and an elliptic curve E defined over it (E(Fp)).
- After that, they publicly pick a random base point B belongs E.
- In third stage, Alex chooses a secret random integer e. Alex then computes eBεE. Consequently, transmit to Benny.
- A secret random integer d is selected by benny. Benny then computes dBεE. And send it to Alex.
- Subsequently eB and dB are public and e and d are secret.
- Alex computes the secret key edB = e(dB).
- Benny computes the secret key edB = d(eB).

Example:
Step 1: Set public parameters
Curve type: F (p), Curve Size: Small, Domain parameters: a=3,b=15,p=31, generator G= (1,9)
Step 2: Choose Secrets
Alice= 11, Bob=14
Step 3: Generate shared keys
Secret key (d): Q=d*G , Alice=(1,22), Bob= (5,0)
Step 4: Exchange shared keys
Step 5: Generate common key
Key = sA*QB and key=sB*QA S= (5,0)

2.4 Elliptic Curve Digital Signature Algorithms (ECDSA)

Initially, the Elliptic Curve Digital Signature Algorithm was proposed in 1992 by Scott Vanstone. Three kinds of algorithm are derived from ECDSA as follows: key generation, signing, and verification. Algorithm ECDSA is the elliptic curve analogue of the Digital signature algorithm. To resolve the issues of ECDLP arise from the hardness of ECDSA. The main benefit of ECDSA is to achieve the same security level as with DSA, however with smaller keys. By using smaller keys can also be evaluated more rapid calculations and smaller public keys to pass around. A public and private key utilize to perform the signing process and verification process by computing the key generation algorithm. The signing procedure is completely executed to generate the actual digital signature. At last procedure, the verification process controls or performs to prove the authenticity of the signature. In ECDSA, that has a alternative approach of the Digital Signature Algorithm (DSA) that works on elliptic curve groups. A signed message is transmitted from A to B and they have to agree up on Elliptic Curve domain parameters. A private key dA and a public key QA = dA * G where G is the generator point, an elliptic curve domain parameter [11]. The algorithm is described by the following steps.

ECC Domain Parameters

The elliptic curve domain parameters are determined a finite field over arithmetic operations for utilizing these public key cryptographic schemes. ECC domain parameters over F_q (where F_q is either F_p and F_{2^m}) are a septuple:

$$T = (q,FR,a,b,G,n,h)$$

Consisting of a number q specifying a prime power ($q = p$ or $q = 2^m$), an indicator FR (field representation) of the method used for representing field elements ε F_q, two field elements a and b ε F_q , that specify the equation of the elliptic curve E over F_q.

ECDSA Key Generation

The user A follows these steps where p is a large prime:

- Select a random integer d ∈ [1, n - 1].
- Compute Q = d x P.

- The public and private keys of the user A are Q and d, respectively.

The other parties can check if the public key is valid by;

- Checking that $Q \neq 0$.
- Checking that x_Q and y_Q are properly represented elements of F_q.
- Checking that Q is on the elliptic curve defined by a and b.
- Checking that $n_Q = Q$.

If any of these checks fail the public key Q is invalid, otherwise Q is valid. The following procedure describes how to generate the signature.

ECDSA Signature Generation

The user A signs the message m using the following steps

- Select a pseudorandom integer $k \in [1, n - 1]$.
- Compute k x P = (x_1, y_1) and r = x_1 mod n.
  If $x_1 \in$, $GF(2^k)$, it is assumed that x_1 is represented as a binary number.
  If r = 0 then go to Step 1.
- Compute k^{-1} mod n.
- Compute $s = k^{-1}(H(m) + d \bullet r) \bmod n$.
  Here H is the secure hash algorithm SHA-1.
  If s = 0 go to Step 1.
- The signature for the message m is the pair of integers (r, s).

ECDSA Signature Verification

The user B verifies A's signature (r, s) on the message m by applying the following steps:

- Verify that r and s are integers in the interval [1,n-1].
- Compute $c = s^{-1}$ mod n and H(m).
- Compute $u_1 = H(m) \bullet c \bmod n$ and $u_2 = r \bullet c \bmod n$.
- Compute u_1 x P + u_2 x Q = (x_0, y_0) and v = x_0 mod n.
- Accept the signature if v = r.

Example

- **ECDSA Key Generation**

Signature originator: parkavi parkavi
Domain parameters to be used 'EC-prime239v1':
Chosen signature algorithm: ECSP-DSA with hash function SHA-1
Size of message M to be signed: 5 bytes
Bit length of c + bit length of d = 477 bits
Message m = "INDIA"
00000 49 4E 44 49 41 INDIA
Elliptic curve E described through the curve equation: y^2 = x^3 + ax + b (mod p) :
a =
83423532389192164791648750360308885314476597252960362792450860609699836
b =
738525217406992417348596088038781724164860971797098971891240423363193866
Private key = 1521284887
Public key W=(Wx,Wy) (W is a point on the elliptic curve) of the signature originator:
Wx =
12247008326168069641975071450044725953573805277085576222307449754331974
Wy =
13684505523169076698381231902079003781912363230473532860089313480758778

Calculate a 'hash value' f (message representative) from message M, using the chosen hash function SHA-1.
f =
856854001393991768291128146113162782518412554266

- ECDSA SIGNATURE as follows:

G has the prime order r and the cofactor k (r*k is the number of points on E):
k = 1
Point G on curve E (described through its (x,y) coordinates):
Gx =
110282003749548856476348533541186204577905061504881242240149511594420911
Gy =
869078407435509378747351873793058868500210384946040694651368759217025454
r =
883423532389192164791648750360308884807550341691627752275345424702807307
The secret key s is the solution of the EC discrete log problem W=x*G(x unknown)

S=0X23017D41791ABDC1867EADB3C88B623DA0A4CEB6C4160E063A27B2504DD

Signature:
c =
362820935521803248322143435026077995999480648951673113245353131958670779
d =
748048045448572568788816039919981279778783010640192427515171177419114296

- ECDSA VERIFICATION as follows:

If c or d does not fall within the interval [1, r-1] then the signature is invalid:
c and d fall within the required interval [1, r-1].
Calculate the number h = d^(-1) mod r:
h =
4293383948764973994857292206205978050702175550742905649742351881512371
Calculate the number h1 = f*h mod r:
h1 =
236463806820815583367739817792232528163119328411731596206743501995551606
Calculate the elliptic curve point P = h1 G + h2 W
Calculate the number h2 = c*h mod r:
h2 =
264516349788480196670041137226853035014476479286199650036313546186557595
(If P = (Px, Py) = (inf, inf) then the signature is invalid):
Px =
362820935521803248322143435026077995999480648951673113245353131958670779
Py =
694333078325502133912962052980607990720812603938058098391829104469112407
Convert the group element Px (x co-ordinates of point P on elliptic curve) to the number i:
i =
362820935521803248322143435026077995999480648951673113245353131958670779
Calculate the number c' = i mod r:

c' =
362820935521803248322143435026077995999480648951673113245353131958670779
If c' = c then the signature is correct; otherwise the signature is invalid:

III.EXPERIMENTAL RESULTS

Performance and analysis measurements has utilized on the Amazon EC2 environment. The Nimbus toolkit installed and affords an infrastructure as a cloud service to its client through Amazon EC2 WSDL or WSRF-based web service APIs. Nimbus is free software as well as subject to the requirements and open-source software of the Apache License version 2. Nimbus Infrastructure is an open source S3-compatible or EC2 Infrastructure-as-a-Service. Particularly, the main targeting features of interest to the scientific community such as best-effort allocations, batch schedulers, proxy credentials, etc. Our performance result has some values for utilizing all the Single-Job benchmarks and in particular time. Optimizations, tuning the benchmarks process were compiled using JAVA command-line arguments. Moreover, we did not use any instance-dependent optimizations or additional architecture.

Moreover, an ECC or a RSA standard server certificates are configured by using a SSL handshake between a server and a client. After that the test methodology is planned to determine the relative differences. The primary difference is articulated because of the public-key cryptographic algorithms as considered in ECC or RSA based algorithms. For that reason, the key exchange is performed on the option of ephemeral ECDH that have to keep and forward the secrecy on that it affords and we do see a popular move towards this as well performed.

The SSL handshake also includes and completes the operations which are identified in the table on Public Key Cryptographic Operations. In this environment model, it has reputation gaining and is simply run the tests available public information to enable the reader to repeat same tests: Amazon EC2. This kind of the test is simultaneously loaded to the server by running the same transaction repetitively through multiple clients and gathering latency (response time) and throughput at the client desktop. It enables setup on the Red Hat Enterprise Linux Server release 6.3 64 bit with our application, High-CPU Extra Large Instance (c1.xlarge) and the Linux.

By applying test data the security algorithms is evaluated in terms of the execution time required to store or retrieve the text data at cloud. The Simulation program accepts inputs: Algorithm and data block. Subsequent to a successful execution i.e. encryption and decryption process generate an efficient result. After the successful encryption/decryption process the analytical table is created. To make sure that all the data are processed in the right way. Basically, it is depend upon execution time (Encryption and Decryption time) as parameters.

A cryptographic technique depends a lot on the size of the key used for security purpose. The Encrypt/Decrypt algorithm will be known to all. This algorithm is always a better choice to have a big key size and also we should keep in mind the computational load after we increase the key size. *ECC based algorithms provide* helps using a lesser key size compared to other cryptographic technique and still holds at high level ***security.*** The key length of our implementation we have used a 160 to 192 bit, which is quit better to protect against naive attack. The key length is increased for better security by using the encryption and decryption process. The ECC, RSA, ECDH and ECDSA algorithms can provide to determine the length of the encryption keys and an arbitrary level of security used for each algorithm. Tables represent the required key length using different encryption algorithms in order to complete a level of security similar to the RSA key length provided by 1024-bit RSA encryption. The times for key generation, signature, and verification algorithms have computed with comparable key sizes for RSA, ECC, ECDH and ECDSA. ***The results of the report showed that ECDSA outperformed RSA in both key and signature generations***. However, ECDSA was capable to verify messages much faster than RSA. The key sizes ranged from 1024 to 15360 bits for RSA and 163 to 571 bits for ECDSA algorithms. In ***ECDSA key generation are consistently faster than those of RSA.*** By the last comparison, RSA has taken a total of 1142.5455 seconds while ECDSA lasted 315.5778 seconds, significantly faster. Meanwhile, the signature generation had slightly different results. RSA started out by executing faster than ECDSA. However, as the bit sizes for each increased, ***RSA has shown to slow down as ECDSA sped up*** and surpassed its counterpart on the final execution. Finally, with ***signature verification, ECDSAs times are considerably quicker than RSAs,*** times and barely increased as the size of key lengths grew.

Table 1: Time Complexity (MS) Comparison of RSA and ECC based Algorithms

Time Complexity(MS)				
Key size (RSA:ECC:ECDH :ECDSA)	**RSA**	**ECC**	**ECDH**	**ECDSA**
1024:163:128:163	212.4521	149.6712	125.4155	114.3564
2048:224:160:233	542.4123	237.6034	189.2541	125.2457
3078:256:192:283	678.9623	456.0023	242.2564	179.2563
7680:384:224:409	817.3245	779.2342	369.1248	298.5447
15360:521:256:571	1142.5455	854.5878	410.2565	315.5778

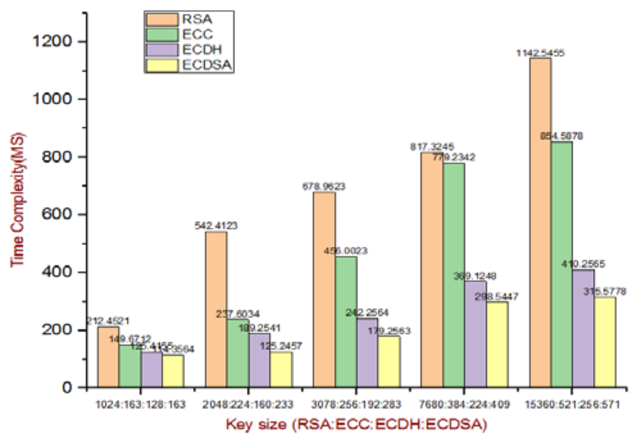


Figure 1: Time Complexity (MS) Comparison of RSA, ECC, ECDH and ECDSA

Table 2: Execution Times for Key Generation and Encryption (MS) Comparison of RSA and ECC based Algorithms

Execution Times for Key Generation and Encryption(MS)				
Key size (RSA:ECC:ECDH :ECDSA)	**RSA**	**ECC**	**ECDH**	**ECDSA**
1024:163:128:163	54.6534	48.2548	46.8545	40.8454
2048:224:160:233	78.2545	55.0485	49.5455	47.4748
3078:256:192:283	98.3523	87.8455	59.0174	56.4854
7680:384:224:409	112.2545	96.5452	69.1484	67.5454
15360:521:256:571	153.8767	106.0457	78.1254	73.8458

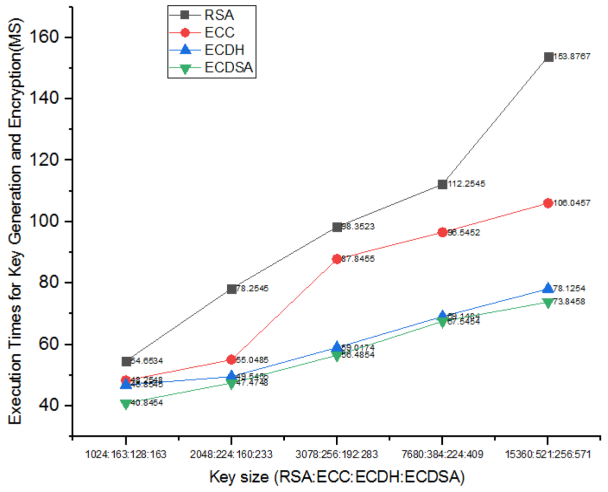


Figure 2: Execution Times for Key Generation and Encryption (MS) Comparison of RSA, ECC, ECDH and ECDSA

Figure 1 and Table 1 shown Time Complexity of Security Algorithms (RSA, ECC, ECDH and ECDSA), Figure 2 and 3, Table 2 & 3 shown, Encryption and decryption performance for the various algorithms are difficult to measure and are heavily influenced by system architecture and software/hardware optimizations. ***ECC offers dramatically superior key pair generation performance compared to RSA,*** with the large primes generated for RSA requiring several orders of magnitude more time when compared to a much smaller ECC key, especially at RSA bit lengths of 1024 and above RSA encryption is generally slightly lesser than ECC, while ***ECC decryption may be several times faster than RSA,*** although both are generally efficient enough not to provide a practical system bottleneck. The ECDH and ECDSA method is assumed to offer a similar processing time as RSA due to similarities in algorithm implementation but will likely take longer due to the multiple exchanges involved.

Table 3: Execution Times for Decryption (MS) Comparison of RSA and ECC based Algorithms

Execution Times for Decryption(MS)				
Key size (RSA:ECC:ECDH: ECDSA)	**RSA**	**ECC**	**ECDH**	**ECDSA**
1024:163:128:163	56.3429	49.8785	44.2645	39.4587
2048:224:160:233	76.2545	70.0124	58.1247	46.2147
3078:256:192:283	95.6955	89.0525	61.2565	57.9876
7680:384:224:409	106.2541	98.0214	97.7249	79.6589
15360:521:256:571	123.5455	101.4521	99.7854	89.6987

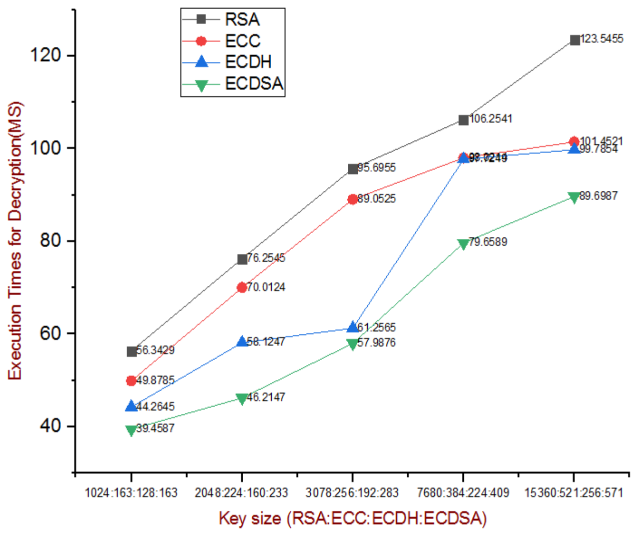


Figure 3: Execution Times for Decryption (MS) Comparison of RSA, ECC, ECDH and ECDSA

Table 4: Execution Times for Signature verification (MS) Comparison of RSA and ECC based Algorithms

Execution Times for Signature verification(MS)				
Key size (RSA:ECC:ECDH: ECDSA)	**RSA**	**ECC**	**ECDH**	**ECDSA**
1024:163:128:163	37.0523	31.9875	29.5457	24.2568
2048:224:160:233	44.2545	39.5784	34.9854	31.5896
3078:256:192:283	56.2556	43.8795	41.2578	39.4112
7680:384:224:409	69.6723	57.9821	52.6541	43.8854
15360:521:256:571	110.2443	73.6942	69.8754	55.9558

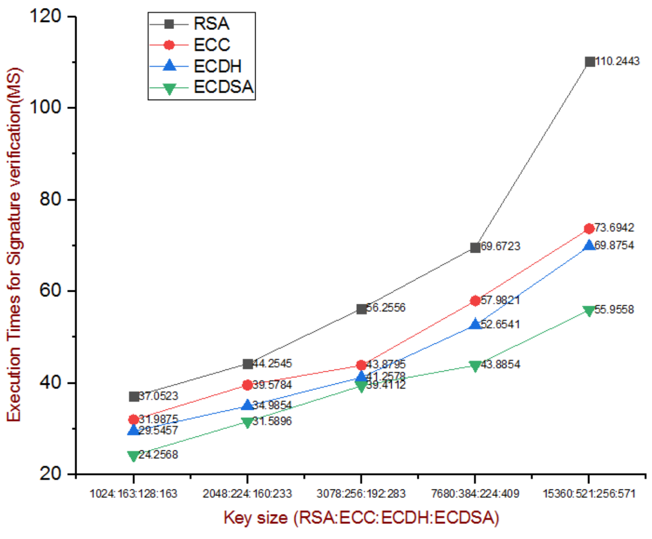


Figure 4: Execution Times for Signature verification (MS) Comparison of RSA, ECC, ECDH and ECDSA

Figure 4 and Table 4 shown Signature Verification of algorithms and key pair generation algorithm of **ECDSA** needs a random number to be generated. Using this random number as seed private keys is generated. Similarly secret integer 'K' generated during signature verification algorithm should also be random in nature. An attacker can exploit this vulnerability if the algorithm used to generate the random number is not cryptographically secure i.e. it should be unpredictable. So probability of any given value being selected should be very small. As a future scope of our proposed work cryptographically secure random number should be included while generating private keys.

IV.CONCLUSION

Encryption and decryption algorithms play an important role in data security on cloud. Various encryption algorithms have been proposed to make cloud data secure, vulnerable and gave concern to security issues and challenges. In this paper the comparisons have been made between **RSA** and **ECC** Based algorithms (**ECC , ECDH** and **ECDSA)** to find the which one is best security algorithm, which has to be used in cloud computing for making cloud data secure and not to be hacked by attackers. According to the experimental results **ECDSA** is best for the remaining algorithms such as **RSA, ECC** and **ECDH.** The future scope of this work is to find out an efficient proposed novel algorithm to make the more secure than **ECDSA**.

V.REFERENCES

[1]. Garfinkel, S.L; “Public Key Cryptography”, Computer, IEEE, Volume: 29, Issue: 6, June 1996.

[2]. Periyanatchi S, Chitra.K. [2015] Analysis on Data Security in Cloud Computing-A Survey. International Conference on Computing and Intelligence Systems 04:1281 – 1284.

[3]. CharanjeetKaur et al. [2015] Data Security Algorithms In Cloud Computing: A Review. International Journal For Technological Research In Engineering 2:372– 375.

[4]. Sana Belguith et al. [2015] Enhancing Data Security in Cloud Computing Using a Lightweight Cryptographic Algorithm. The Eleventh International Conference On Autonomic and Systems. 98– 103.

[5]. Tembhurne S et al. [2015] An Improvement In Cloud Data Security That Uses Data Mining. International Journal of Advanced Research in Computer Engineering & Technology 4: 2044– 2049.

[6]. Nikhitha K, Navin K S. [2015] A Survey On Various Encryption Techniques For Enhancing Data Security In Cloud. International Journal of Advanced Research Trends in Engineering and Technology 194– 197.

[7]. S. Maria Celestin Vigila , K. Muneeswaran “Implementation of Text based Cryptosystem using Elliptic Curve Cryptography”, IEEE Sep-2009, pp. 82-85.

[8]. Abhuday Tripathi, and Parul Yadav, “Enhancing Security of Cloud Computing using Elliptic Curve Cryptography, International Journal of Computer Applications, 57(1), 2012, 0975-8887.

[9]. Sherif El-etriby, Eman M. Mohamed,Modern Encryption Techniques for Cloud Computing,proceedings of the informatics and systems 8th international conference(page:cc-1-cc-6 year :2012 ISBN: 978-1-4673-0828-1)

[10]. Mandeep Kaur, Manish Mahajan, “Using encryption Algorithms to enhance the Data Security in Cloud Computing”, International Journal of Communication and Computer Technologies,Vol.12, Issu.3,pp 56-59,2013

[11]. Julio Lopez and Ricardo Dahab- Fast Multiplication on Elliptic Curves over GF(2m) without Precomputation, Springer.

ABOUT THE AUTHORS

D.Pharkkavi received her M.Phil Degree from Tiruvalluvar University, Vellore in the year 2013. She has received her M.C.A Degree from Anna University, Chennai in the year 2012. She is pursuing her Ph.D (Full-Time) Degree at Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India. Her areas of interest include Cloud Computing and Mobile Computing.

Dr.D.Maruthanayagam received his Ph.D Degree from Manonmaniam Sundaranar University, Tirunelveli in the year 2014. He received his M.Phil Degree from Bharathidasan University, Trichy in the year 2005. He received his M.C.A Degree from Madras University, Chennai in the year 2000. He is working as HOD Cum Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India. He has above 15 years of experience in academic field. He has published 4 books, 26 papers in International Journals and 28 papers in National & International Conferences so far. His areas of interest include Computer Networks, Grid Computing, Cloud Computing and Mobile Computing.