Space Complexity Analysis of RSA and ECC Based Security Algorithms in Cloud Data

[1]D.Pharkkavi, [2]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.

***Abstract*:** Cloud computing is an important development trend in information technology all over the world. Nowadays, the cloud data security technique exploits the symmetric encryption and asymmetric encryption algorithms within the capability of the stronger authentication techniques. A major risk of data security in cloud computing environment becomes a serious problem by reason of the data which is stored diversely over the cloud. Both the data security and privacy are the two main characteristics of cloud information technologies for user's concern. We discuss in this paper, a number of existing techniques used to provide security in the field of cloud computing on the basis of different parameters. It will be helpful to improve assure the security of data storage in a cloud environment.

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

*****

I. INTRODUCTION

Today's, the cloud computing is a well-known technology to improve data security and privacy. Companies such as Microsoft, Google and Amazon are improving or developing the services provided for their user's requirements. Privacy acts that are exploit in or out of date, after which are not protecting the private information of user in the cloud environment. For the reason that, they are not applicable to three parties such as cloud provider, cloud service user, cloud service provider. Privacy risk becomes worse when applications are present in multiple locations. The non-adequate security characteristics and measures of the cloud service providers such as audit, control, confidentiality, and data integrity availability have been added [1]. In cloud systems, security issue is a barrier for users to adapt into cloud systems. Afterward, an application runs in the public domain or beyond the firewall and then there occurs security concerns and consciousness. In cloud computing, the consumers can allow to access resources online at anywhere or anytime via Internet without controlling the original resources problems such as technical and physical management.

Cloud computing resources are accessed in the form of scalable and dynamic manner. Enterprise control loss opposed to particular technical challenge by the cloud security with the significant difference. The application of security, infrastructure and platform is under provider's control. In cloud based application access control is important [5]. Responsibility of the providers, they affords the physical security, virtualization security and environmental security depends upon IaaS offering by Amazon's EC2, the security responsibility of the consumers, is up to perform with operating system, data and application. For example of Salesforce.com's Customer Resource Management (CRM), which is a SaaS offering. The complete responsibility lies with the provider which means that it will take care of environmental, physical, data and application security controls and this will relieve the customer. Service provider and customer depend on keeping in the view of the service model have total responsibility of security of infrastructure in cloud computing [6]. The security controls in cloud computing are same as in any other IT environment, due to different technologies and deployment models are utilized to afford cloud computing services that may pose some different risks to the organization [7]. The cloud service providers have to manage the security and also to deliver diverse services to several users and then they take steps to enhance security the services become more rigid. It may pose some different risks to organization and these risks arise mostly at the network layer of security controls.

Data Integrity: The consumer desires that [8]

1. To protect the data integrity by using fine-grained access control and protection from intruders or hackers and single sign-on or sign-off.
2. It may allow to access cloud resources with security protocols such as Secure Sockets Layer (SSL) or HTTPS with the compliance checking and the security auditing control.
3. Shared datasets are protected against from copyright violation or deletion, and malicious alteration.

Data Theft: To encrypt the data using one possible solution [9]. Personal firewalls and Shared datasets are securely protected from ActiveX Applets, Java, and JavaScript with established VPN channels between resource sites and cloud clients [8].

Infected Application: The vendor must have access to the servers with the intention that they can check whether if any malicious user has uploaded any infected application. In that case, they may take the crucial actions to avoid any inconvenience to the customer.

Privacy Issues: The provider and client must have an equal privacy policy for performing better results. The provider has been assigned to every user an access control mechanism indicating when and who is available to access the data. Clients also want to look or prefer at all the access log of the vendor employees and also of their employees [10].

Compliance: Compliance refers to the responsibility of an organization to work under a specific agreement with established standards, laws and regulations. Compliance becomes a c omplex issue for cloud service provider due to varying privacy and security laws administrated in different countries [11].

Governance: Governance means to have proper control over measures, principles and policies for IT service achievement [11]. If governance is compromised after that the measures and policies for security can be ignored. IT infrastructure manages a complex set of software and hardware environments. These services are supplied to a customer with an authorization level of service level.

1) ***Law and Regulations***: Laws like HIPAA and SOX etc. involve the customer to be responsible for utilizing the privacy and security of data hosted in the cloud. Although cloud service providers are becoming responsive to different laws and regulations, which may store data in specific control and apply needed protection for privacy and security.
2) ***Data Location***: Nowadays, the data location is one of the most essential compliance risks faced by every organization [12]. In that case, the data center housed within organization premises, when the data is transparent to protect the security controls with the data location. In the typical cloud computing environment, the data is stored in numerous physical locations and data location is unknown to the service customer.

Trust: In a cloud environment, an organization handles control over many aspects of security to protect by placing its trust in the cloud service provider [11] [12]. An organization brings with the intrinsic level of risk while performing data is being stored outside the physical boundaries [11]. The Insider access issue or threats include theft information, fraud access, and information resources sabotage as equally true in the cloud environment, and it’s apart from causing an incident intentionally or unintentionally happens as possible. While moving organizational data covert into the cloud do not only broaden the domain of threat from organizational staff but also from other cloud customers utilizing and performing with sharing resources and cloud services such as virtual machine instances for computational requirements in cloud computing.

Data Ownership: Cloud service provider should not be given all rights to use or alter the data for its own purpose or gain. Data Ownership is significantly handled that an organization holds possession over all its data.

Data Protection: Data is stored in a shared environment and that the shared data is located with other customer’s data in cloud. To keep data against or away from unauthorized user access control as well encryption is the only choices and data types that are stored in the cloud. When access control mechanism is typically identity based, and encryption remains the only way to protect and assure the data.

Identity and Access management: Illegal identification and access prevention have also become one of main concern for cloud service providers and to move toward adopting cloud handles data sensitivity problems and privacy issues. Nowadays, SAML standard is being used by the number of cloud service providers to manage users in the cloud.

In cloud computing, the security issues have categorized into two levels. They are given below,

Security issues faced by cloud providers then cloud: The cloud provider should protect the data and application of the cloud users. The cloud provider guarantees that the infrastructure is secured and protected against from unauthorized access.

Security problems faced by customers: Majority of security issue regard as virtualization that can be properly managed, configured and secured. At the same time as the customer should authorize that the provider has makes the proper security measures to protect their infrastructure.

II .SECURITY ALGORITHMS

2.1. RSA

In RSA schema, integer performs between the interval [0, n-1] such as block cipher, original message and cipher message. RSA is broadly used algorithm in various fields such as bank, e-commerce, military and so on.In which the encrypted message and original message are represented h*h square matrices in another schema. For encryption and decryption order, they don’t have any restriction and also consider as more efficient, dynamic and scalable [4]. For the above security purpose, the hardware implementation of RSA schema make use of the modular exponentiation [5] and also provide security and facilitate 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 and consists of plaintext and ciphertext in the form of integers within 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 procedure is completed in three steps:

- Key generation
- Encryption
- Decryption

Key Generation: In key generation, two prime numbers are considered (i.e.) p and q and consists of public key and a private key. The public key is well-known to everybody. Calculate the value of n and choose a random encryption key e evaluates the gcd and that should be equal to 1. Subsequently, find out the decryption key d. At last evaluate the public key and private key in an effective manner. The plain text is encrypted in blocks, each block contain a binary value less than that number n i.e., for block size i bits, $2^i<n<2^{i+1}$.

- Input: None
- Calculations: Choose two comparatively 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:

The encryption process represents a plaintext in the form of numbers modulo n to obtain cipher text C from plaintext M is very trouble-free. It can be formulated as: $C=M^e \bmod n$

If C = cipher text
D = private key
E = public key
M = message text
The file can be encrypted by transmitting a symmetric file encrypted key (FEK) concurrently asymmetric public key will be automatically generated and then both are combined to form an encrypted FEK with a header file.

- Input: Integers n, e, M
- Integer representation of the plain text is M
- Let C be Evaluated as 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.
- Let D decrypted text evaluated such that $D=(C^d \text{ Mod } n)$

- Output: the decrypted message represent as D.
- Public Key **Encryption** : {e, n}
- Private Key **Decryption**: {d, n}

Example:
i) Prime Number P = 193, Q = 131.
ii) RSA Modules N= 193 x 131 = 25283
$\phi(n) = (193-1).(131-1) = 24960.$
iii) Public key e = 2^16+1= 65537.
iv) Private key $d \equiv e^{-1} \pmod{24960} \equiv 15233$.
v) Message M= "The Republic of India is a country in Asia. At the center point of South Asia is Indi and it has more than 1.2 billion people. In South Asia, India is the seventh largest country in the world by area wise and also the most populous democracy in the world. New Delhi is the capital of India and coastline of India is about of 7,517 km (4,671 mi) long. India is a peninsula region, bound with the Bay of Bengal in the east region, the Arabian Sea on the west region and Indian Ocean in the south region. In the world survey, India has the third largest military force and is also a nuclear weapon state. India has seven neighbor countries followed by: Myanmar in the east, Bhutan and Bangladesh in the north-east, Pakistan in the north-west, China and Nepal in the north, and Sri Lanka, an island, in the south.
vi) Encryption E **(M)** ≡ **M** $^{\mathbf{65537}}$**(mod 25283**).
Vii) Decryption D **(M)** ≡ **M**$^{\mathbf{15233}}$ **(mod 25283**).
Viii) Benny sends Alex the message **"sample.txt"** files as follows:

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

T # h # e # # R # e # p # u # b # l # i # c # # o # f # # I # n # d # i # a # # i # s # # a # # c # o # u # n # t # r # y # # i # n # # A # s # i # a # . # # I # t # # i # s # # a # t # # t # h # e # # c # e # n # t # e # r # # o # f # # S # o # u # t # h # # A # s # i # a # . # # I # n # d # i # a # # h # a # s # # m # o # r # e # # t # h # a # n # # 1 # . # 2 # # b # i # l # l # i # o # n # # p # e # o # p # l # e # , # # w # h # i # c # h # # i # s # # t # h # e # # s # e # c # o # n # d # # l # a # r # g # e # s # t # # p # o # p # u # l # a # t # i # o # n # # i # n # # t # h # e # # w # o # r # l # d # . # # I # t # # i # s # # t # h # e # # s # e # v # e # n # t # h # # l # a # r # g # e # s # t # # c # o # u # n # t # r # y # # i # n # # t # h # e # # w # o # r # l # d # # b # y # # a # r # e # a # # a # n # d # # t # h # e # # l # a # r # g # e # s # t # # c # o # u # n # t # r # y # # i # n # # S # o # u # t # h # # A # s # i # a # .

Numbers input in base 10 format.
084 # 104 # 101 # 032 # 082 # 101 # 112 # 117 # 098 # 108 # 105 # 099 # 032 # 111 # 102 # 032 # 073 # 110 # 100 # 105 # 097 # 032 # 105 # 115 # 032 # 097 # 032 # 099 # 111 # 117 # 110 # 116 # 114 # 121 # 032 # 105 # 110 # 032 # 065 # 115 # 105 # 097 # 046 # 032 # 073 # 116 # 032 # 105 # 115 # 032 # 097 # 116 # 032 # 116 # 104 # 101 # 032 # 099 # 101 # 110 # 116 # 101 # 114 # 032 # 111 # 102 # 032 # 083 # 111 # 117 #

116 # 104 # 032 # 065 # 115 # 105 # 097 # 046 # 032 # 073 # 110 # 100 # 105 # 097 # 032 # 104 # 097 # 115 # 032 # 109 # 111 # 114 # 101 # 032 # 116 # 104 # 097 # 110 # 032 # 049 # 046 # 050 # 032 # 098 # 105 # 108 # 108 # 105 # 111 # 110 # 032 # 112 # 101 # 111 # 112 # 108 # 101 # 044 # 032 # 119 # 104 # 105 # 099 # 104 # 032 # 105 # 115 # 032 # 116 # 104 # 101 # 032 # 115 # 101 # 099 # 111 # 110 # 100 # 032 # 108 # 097 # 114 # 103 # 101 # 115 # 116 # 032 # 112 # 111 # 112 # 117 # 108 # 097 # 116 # 105 # 111 # 110 # 032 # 105 # 110 # 032 # 116 # 104 # 101 # 032 # 119 # 111 # 114 # 108 # 100 # 046 # 032 # 073 # 116 # 032 # 105 # 115 # 032 # 116 # 104 # 101 # 032 # 115 # 101 # 118 # 101 # 110 # 116 # 104 # 032 # 108 # 097 # 114 # 103 # 101 # 115 # 116 # 032 # 099 # 111 # 117 # 110 # 116 # 114 # 121 # 032 # 105 # 110 # 032 # 116 # 104 # 101 # 032 # 119 # 111 # 114 # 108 # 100 # 032 # 098 # 121 # 032 # 097 # 114 # 101 # 097.

ix) Encryption into ciphertext c[i] = m[i]^e (mod N)

17092 # 01455 # 09557 # 23528 # 02873 # 09557 # 18447 # 07644 # 13093 # 12050 # 11299 # 20750 # 23528 # 17778 # 00208 # 23528 # 13273 # 11880 # 15348 # 11299 # 22635 # 23528 # 11299 # 07344 # 23528 # 22635 # 23528 # 20750 # 17778 # 07644 # 11880 # 03264 # 24664 # 08227 # 23528 # 11299 # 11880 # 23528 # 05653 # 07344 # 11299 # 22635 # 17567 # 23528 # 13273 # 03264 # 23528 # 11299 # 07344 # 23528 # 22635 # 03264 # 23528 # 03264 # 01455 # 09557 # 23528 # 20750 # 09557 # 11880 # 03264 # 09557 # 24664 # 23528 # 17778 # 00208 # 23528 # 07227 # 17778 # 07644 # 03264 # 01455 # 23528 # 05653 # 07344 # 11299 # 22635 # 17567 # 23528 # 13273 # 11880 # 15348 # 11299 # 22635 # 23528 # 01455 # 22635 # 07344 # 23528 # 02401 # 17778 # 24664 # 09557 # 23528 # 03264 # 01455 # 22635 # 11880 # 23528 # 21100 # 17567 # 04296 # 23528 # 13093 # 11299 # 12050 # 12050 # 11299 # 17778 # 11880 # 23528 # 18447 # 09557 # 17778 # 18447 # 12050 # 09557 # 11314 # 23528 # 23472 # 01455 # 11299 # 20750 # 01455 # 23528 # 11299 # 07344 # 23528 # 03264 # 01455 # 09557 # 23528 # 07344 # 09557 # 20750 # 17778 # 11880 # 15348 # 23528.

Alex decrypts the message by computing, Decryption into plaintext m[i] = c[i]^d (mod N)

03775 # 21141 # 01837 # 25072 # 19085 # 01837 # 14973 # 01468 # 18497 # 11278 # 17282 # 19206 # 25072 # 15655 # 08507 # 25072 # 07676 # 00686 # 02417 # 17282 # 20512 # 25072 # 17282 # 01940 # 25072 # 20512 # 25072 # 19206 # 15655 # 01468 # 00686 # 18511 # 22541 # 23281 # 25072 # 17282 # 00686 # 25072 # 09899 # 01940 # 17282 # 20512 # 15444 # 25072 # 07676 # 18511 # 25072 # 17282 # 01940 # 25072 # 20512 # 18511 # 25072 # 18511 # 21141 # 01837 # 25072 # 19206 # 01837 # 00686 # 18511 # 01837 # 22541 # 25072 # 15655 # 08507 # 25072 # 21702 # 15655 # 01468 # 18511 # 21141 # 25072 # 09899 # 01940 # 17282 # 20512 # 15444 # 25072 # 07676 # 00686 # 02417 # 17282 # 20512 # 25072 # 21141 # 20512 # 01940 # 25072 # 09349 # 15655 # 22541 # 01837 # 25072 # 18511 # 21141 # 20512 # 00686 # 25072 # 04888 # 15444 # 12402 # 25072 # 18497 # 17282 # 11278 # 11278 # 17282 # 15655 # 00686 # 25072 # 14973 # 01837 # 15655 # 14973 # 11278 # 01837 # 13437 # 25072 # 08611 # 21141 # 17282 # 19206 # 21141 # 25072 # 17282 # 01940 # 25072 # 18511 # 21141 # 01837 # 25072 # 01940 # 01837 # 19206 # 15655 # 00686 # 02417.

2.2. Elliptic curve cryptography (ECC)

Elliptic curve cryptography (ECC) is a cryptographic scheme that uses the properties of elliptic curves to generate cryptographic algorithms. 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:

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

A straight line that intersects the curve within two points and also intersects the curve in a third point that is either on the point or the curve of infinity (also referred to as the neutral element). An additional significant property of elliptic curves is the symmetric over the x-axis that means if you have a point P(x, y) then -P will be (x, -y). By using these properties can describe some useful and interesting arithmetic rules. In case that you have a point A and a point B on an elliptic curve, and you desire to perform an addition operation of these two points. After that, a line draws from A via B, if the line will intersect the curve in a third point that takes it and mirror it over the x-axis and provide the result of the addition.

The main benefit of ECC becomes clear when surveying the security level that keys of different bit sizes provide. ECC based keys generate the physically powerful level of security. On the other hand, the ratios illustrate that the double size of ECC based key but the RSA key size has to be increased more than double that and also observe that escalates for even greater key sizes.

In some case of the RSA, we discuss doubling-up the length of the key saves the performance by a factor of 5-7 [21]. Due to Mark Knight also states that the ***key generation can be a 1,000 times faster with ECC than with RSA.*** A combination of these benefits outcomes such as reduction of network and memory, storage overheads.

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 rand 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 when the signature is not valid move to next step.
- Evaluation: the message digest calculated from the received message using 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

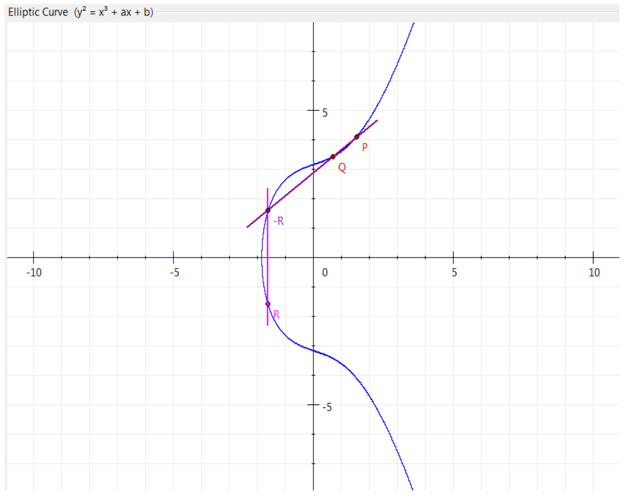
- The plain text M is mapped onto the elliptic curve at a point P.
- Generate a random integer rand between 1 and n-1.
- The cipher text is then encoded as a pair C, where C = [(rand * B),(P + (rand * Bpub)]

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=2, b=10, Curve: $y^2 = x^3 + 2x + 10$, Point P = (1.55|4.1), Point Q = (0.7|3.42),Point R = P + Q = (-1.61|-1.58)

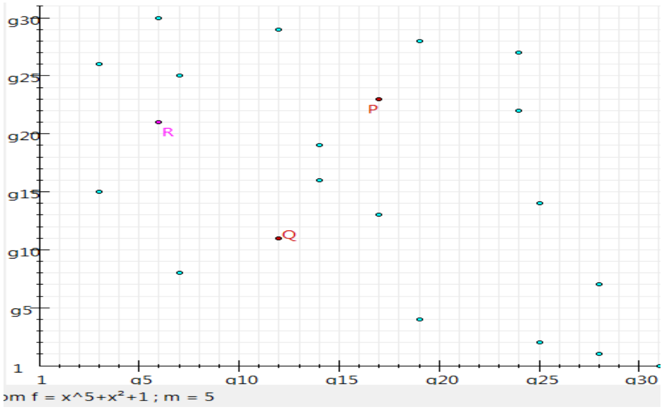


2. Curve Size: Large, Curve Type: F(p), Select curve attributes: ANSI X9.62,Curve: prime192v1, Radix: 16 hexadecimal, Curve attributes: $y^2 = x^3 + 2x + 10$, where
a = fffffffffffffffffffffffffffffffefffffffffffffffc
b = 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
p = fffffffffffffffffffffffffffffffeffffffffffffffff
Base point G: Point P
x = 188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
y= 7192b95ffc8da78631011ed6b24cdd573f977a11e794811
Base point G: point Q
x = 188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
y=7192b95ffc8da78631011ed6b24cdd573f977a11e794811
Point R : R = P + Q
x= dafebf5828783f2ad35534631588a3f629a70fb16982a888
y= dd6bda0d993da0fa46b27bbc141b868f59331afa5c7e93ab

3. Curve Size: Large, Curve Type: F(2^m), Select curve attributes : ANSI X9.62, Curve: c2pnb163v1, Radix : 16 hexadecimal
a = 72546b5435234a422e0789675f432c89435de5242
b = c9517d06d5240d3cff38c74b20b6cd4d6f9dd4d9
m = 163
Base Point P:
x = 00000007 af699895 46103d79 329fcc3d 74880f33 bbe803cb
y= 00000001 ec23211b 5966adea 1d3f87f7 ea5848ae f0b7ca9f

Base point Q:
X=00000007 af699895 46103d79 329fcc3d 74880f33 bbe803cb
Y= 00000001 ec23211b 5966adea 1d3f87f7 ea5848ae f0b7ca9f
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=5, f = x^5+x^2+1,a=1,b=1,Curve: y² + xy = x³ + x² + 1 , Point P = (g17|g23), Point Q = (g12|g11), Point R = P + Q = (g6|g21)



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. By using the shared keys consists of a key or the derived key and also perform to encrypt following 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. A private key is randomly picked integer less then n, if n is the order of the curve and another public key is randomly picked as Q= d*G (G is represent the generator point). Both the sender and receiver have a key pair that consists of a public key and a private key. 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.

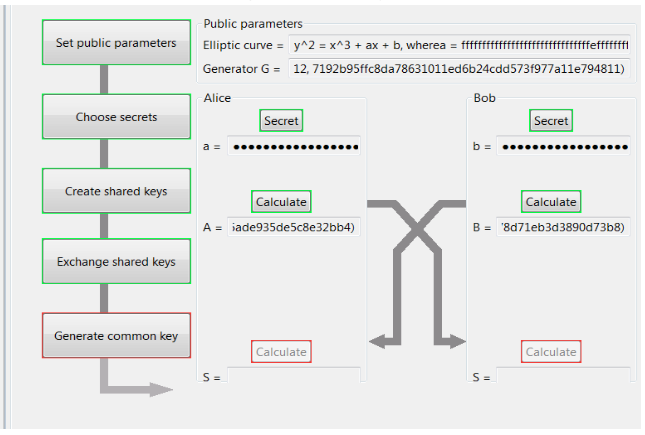
Diffie–Hellman key exchange system Using ECC

- 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:

Curve type: F (p), Curve Size: Large, Domain parameters: a=2, b=10, p=23, generator G=(188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012,7192b95ffc8da78631011ed6b24cdd573f977a11e794811)
Step 2: Choose Secrets
Alex = 5588337381279950393125466200813416822197388
Benny =9138556193173203136734803100548972838505110
Step 3: Generate shared keys
Secret key (d): Q=d*G ,
Alex=(21068c5434005165359f2b036d1fe4a1cc8d04cee073c43a, 45d61eafaa008f328a5578aec2e415ade935de5c8e32bb4)
Benny=(61be9950b088922e7f3ee094b78b776c499a52f2a9c5adcf, 1d5b9728f0d8f88a2086a09839790378d71eb3d3890d73b8)

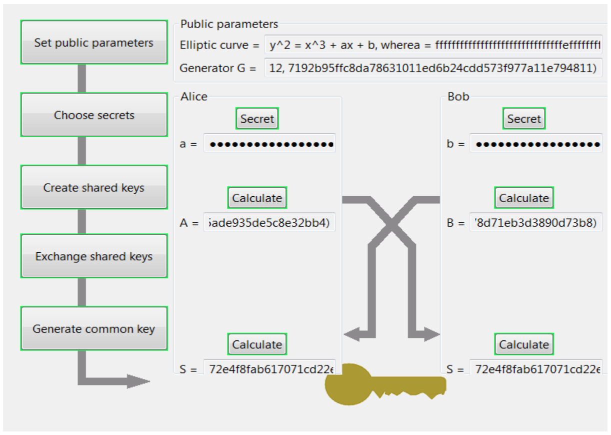
Step 4: Exchange shared keys



Public parameters
Elliptic curve = y^2 = x^3 + ax + b, wherea = ffffffffffffffffffffffffffffffffefffffffi
Generator G = 12, 7192b95ffc8da78631011ed6b24cdd573f977a11e794811)
Alice
a = ••••••••••••••••••
A = ;ade935de5c8e32bb4)
S =
Bob
b = ••••••••••••••••••
B = '8d71eb3d3890d73b8)
S =

Step 5: Generate common key
Key = sA*QB and key=sB*QA

S=
72e4f8fab617071cd22eb053f60f0a646742b55620253116



Elliptic curve = y^2 = x^3 + ax + b, wherea = ffffffffffffffffffffffffffffffffeffffffff

Generator G = 12, 7192b95ffc8da78631011ed6b24cdd573f977a11e794811)

a = ••••••••••••••••• b = •••••••••••••••••

A = ›ade935de5c8e32bb4) B = '8d71eb3d3890d73b8)

S = 72e4f8fab617071cd22ε S = 72e4f8fab617071cd22ε

Specify

2.4 Elliptic Curve Digital Signature Algorithms (ECDSA)

Three kinds of algorithm are derived from ECDSA as follows: key generation, signing, and verification. At first, the Elliptic Curve Digital Signature Algorithm was proposed in the year of 1992 by Scott Vanstone. 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 sent out from A to B and to agree up on Elliptic Curve domain parameters. A private key dA and a public key QA = dA * G where G is represent the generator point, an elliptic curve domain parameter [11]. Following steps briefly explained about this algorithm.

ECC Domain Parameters

The elliptic curve domain parameters determine a finite field number of arithmetic operations for performing these public key cryptographic schemes and ECC domain parameters represent over F_q (where F_q is either F_p and $F_2{}^m$) are a septuple:

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

A number of q specifies a prime power ($q = p$ or $q = 2^m$), an indicator FR (field representation), 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:

- Choose a random integer $d \in [1, n - 1]$.
- Calculate $Q = d \times P$.
- The public key and the private key of the user A are Q and d, respectively.

Other parties know how to 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.
- Confirming that Q is on the elliptic curve named by a and b.
- Proving that $n_Q = Q$.

If it is fail and the public key Q is automatically invalid, otherwise Q is valid. The following steps explain how to produce the signature.

ECDSA Signature Generation

User A signs the message m using the following steps

- Choose a pseudorandom integer $k \in [1, n - 1]$.
- Calculate $k \times P = (x_1, y_1)$ and $r = x_1 \bmod n$.
  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.
- Evaluate $k^{-1} \bmod n$.
- Evaluate $s = k^{-1}(H(m) + d \bullet r) \bmod n$.
  at this time 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

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 $c = s^{-1} \bmod n$ and H(m).
- Compute $u_1 = H(m) \bullet c \bmod n$ and $u_2 = r \bullet c \bmod n$.
- Compute $u_1 \times P + u_2 \times Q = (x_0, y_0)$ and $v = x_0 \bmod n$.
- Accept the signature if $v = r$.

Example:

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: 800 bytes

Bit length of c + bit length of d = 477 bits

File Name = sample.txt

Encrypted Data:

20 54 68 65 20 52 65 70 75 62 6C 69 63 20 6F 66 20 49 6E 64 69 61 20 69 73 20 61 20 63 6F 75 6E 74 72 79 20 69 6E 20 41 73 69 61 2E 20 49 74 20 69 73 20 61 74 20 74 68 65 20 63 65 6E 74 65 72 20 6F 66 20 53 6F 75 74 68 20 41 73 69 61 2E 20 49 6E 64 69 61 20 68 61 73 20 6D 6F 72 65 20 74 68 61 6E 20 31 2E 32 20 62 69 6C 6C 69 6F 6E 20 70 65 6F 70 6C 65 2C 20 77 68 69 63 68 20 69 73 20 74 68 65 20 73 65 63 6F 6E 64 20 6C 61 72 67 65 73 74 20 70 6F 70 75

6C 61 74 69 6F 6E 20 69 6E 20 74 68 65 20 77 6F 72 6C 64 2E 20 49 74 20 69 73 20 74 68 65 20 73 65 76 65 6E 74 68 20 6C 61 72 67 65 73 74 20 63 6F 75 6E 74 72 79 20 69 6E 20 74 68 65 20 77 6F 72 6C 64 20 62 79 20 61 72 65 61 20 61 6E 64 20 74 68 65 20 6C 61 72 67 65 73 74 20 63 6F 75 6E 74 72 79 20 69 6E 20 53 6F 75 74 68 20 41 73 69 61 2E 20 49 74 20 69 73 20 61 6C 73 6F 20 74 68 65 20 6D 6F 73 74 20 70 6F 70 75 6C 6F 75 73 20 64 65 6D 6F 63 72 61 63 79 20 69 6E 20 74 68 65 20 77 6F 72 6C 64 2E 49 6E 64 69 61 20 68 61 73 20 73 65 76 65 6E 20 6E 65 69 67 68 62 6F 75 72 73 3A 20 50 61 6B 69 73 74 61 6E 20 69 6E 20 74 68 65 20 6E 6F 72 74 68 2D 77 65 73 74 2C 20 43 68 69 6E 61 20 61 6E 64 20 4E 65 70 61 6C 20 69 6E 20 74 68 65 20 6E 6F 72 74 68 2C 20 42 68

Elliptic curve E described through the curve equation: y^2 = x^3 + ax + b (mod p) :

a =
883423532389192164791648750360308885314476597252960362792450860609699836

b =
738525217406992417348596088038781724164860971797098971891240423363193866

Private key = 1524894406

Public key W=(Wx,Wy) (W is a point on the elliptic curve) of the signature originator:

Wx =
330620625581488257885526250845089414191075021915398297869755740344684193

Wy =
140782890137067514270411599217713481843211313210160371856876569785276171

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

f =
1350281218722005574507267976653564545028547064787

- 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=
867394451498200440756814680713079325021677452565139760374640511533239520

Signature:

Convert the group element Vx (x co-ordinates of point V on elliptic curve) to the number i:

i =
691511465474034887548580193343725311270936344489552291378763098257452849

Calculate the number c = i mod r (c not equal to 0):

c =
691511465474034887548580193343725311270936344489552291378763098257452849

Calculate the number d = u^(-1)*(f + s*c) mod r (d not equal to 0):

d =
410668840649113958377698162777682846366144595307558067683297720930306317

- 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 =
134939699561555966237295504518629234573113880345070345304610401609566723

Calculate the number h1 = f*h mod r:

h1 =
134662791051587987296931117742237728590953066950899754734898620622440583

Calculate the elliptic curve point P = h1 G + h2 W

Calculate the number h2 = c*h mod r:

h2 =
604971968283495661620588376816607981176025892933600927574724590733196071

(If P = (Px, Py) = (inf, inf) then the signature is invalid):

Px =
691511465474034887548580193343725311270936344489552291378763098257452849

Py =
853501410086344408203973683433688260565292824461894688005891828665918463

Convert the group element Px (x co-ordinates of point P on elliptic curve) to the number i:

i =
691511465474034887548580193343725311270936344489552291378763098257452849

Calculate the number c' = i mod r:

c' =
691511465474034887548580193343725311270936344489552291378763098257452849

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. Nimbus toolkit affords an infrastructure clouds to scientific users. As a cloud service to its client through WSRF-based or Amazon EC2 web service APIs. Nimbus is free software as well as 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 Linux Server, 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. Encryption and decryption process generate an efficient result. After the successful encryption/decryption process the analytical table is created and makes sure all the data processed in the right way and 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 afford* by using a lesser key size compare to other cryptographic technique and still keep at high level of security**.** The key length of the implementation is a 160 to 192 bit that 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***. ECDSA able to verify messages faster than RSA and the key sizes range from 163 to 571 bits for ECDSA algorithm and 1024 to 15360 bits for RSA algorithm. 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. As a final point, with ***signature verification, ECDSAs times are significantly quicker than RSAs,*** times and barely enhanced as the size of key lengths grew.

. **Table 1: 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	102.893	94.505	87.0945	80.3154
2048:224:160:233	127.835	102.74	91.7855	88.1548
3078:256:192:283	149.272	136.98	103.257	98.3754
7680:384:224:409	164.515	147.12	115.388	110.6454
15360:521:256:571	207.477	158.06	126.365	118.1558

In Table 1 and 2, Figure 1 and 2 shown, By using the various algorithms performs with the encryption process, and decryption process that are heavily influenced and difficult to measure through software/hardware optimizations and system architecture. Particularly at RSA bit lengths of 1024 and above RSA encryption is slightly lesser than ECC, while ***ECC decryption can be several times quicker compare than RSA,*** even though both are commonly efficient enough not to provide a practical system bottleneck. The ECDSA and

ECDH technique is assumed to provide a similar processing time as RSA due to similarities in algorithm implementation however will possibly take longer due to the multiple exchanges involved. ***ECC provides dramatically superior key pair generation performance compared to RSA***, with the large primes generated for RSA requiring those orders of magnitude more computation time when compared to a much smaller ECC key.

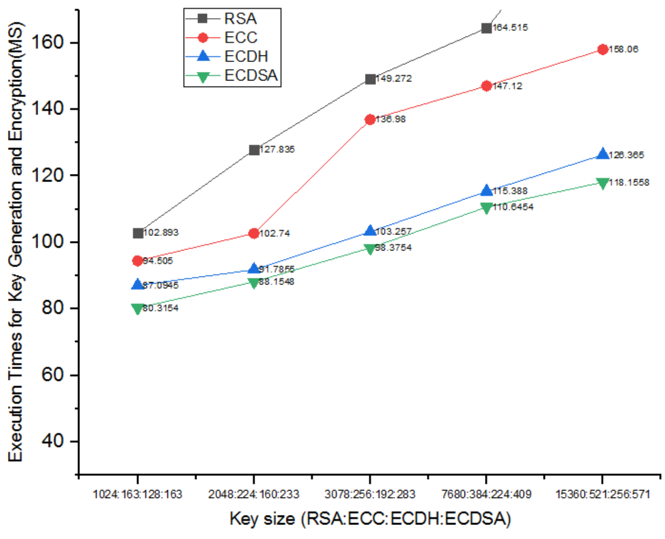


Figure 1: Execution Times for Key Generation and Encryption (MS) Comparison of RSA, ECC, ECDH and ECDSA

Table 2: 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	112.5829	95.0185	78.3045	62.3987
2048:224:160:233	126.7045	111.1324	89.9147	68.6747
3078:256:192:283	140.3555	126.1525	90.7965	79.9676
7680:384:224:409	145.1241	131.1014	125.0149	101.1589
15360:521:256:571	156.6255	130.5121	124.8254	110.7187

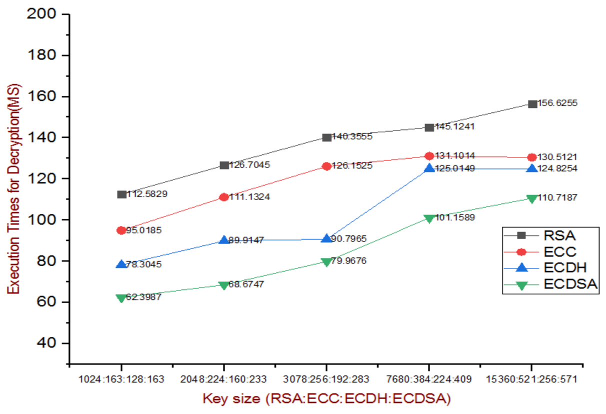


Figure 2: Execution Times for Decryption (MS) Comparison of RSA, ECC, ECDH and ECDSA

Table 3: 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	48.2923	40.4875	35.3057	27.1568
2048:224:160:233	53.6045	46.9784	40.4354	35.0896
3078:256:192:283	63.7156	50.1795	46.3978	43.5112
7680:384:224:409	75.2423	63.1821	57.4841	48.5854
15360:521:256:571	113.9243	77.7942	74.3954	61.2558

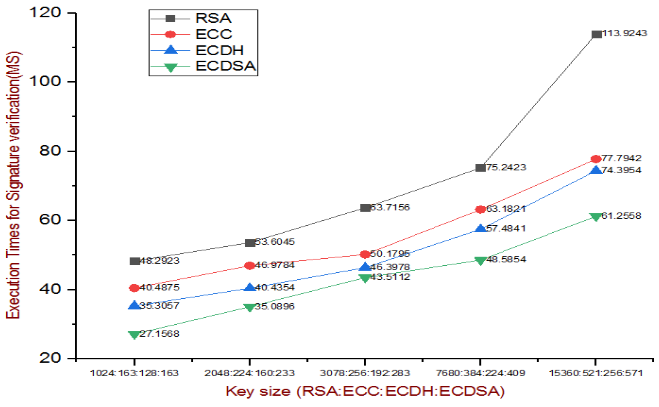


Figure 3: Execution Times for Signature verification (MS) Comparison of RSA, ECC, ECDH and ECDSA

In Table 3 and Figure 3 demonstrate Key Pair Generation algorithm and Signature Verification of algorithms of ECDSA requires a random number to be produced. By using the random number as the private keys are generated. Likewise, the secret integer ‘K’ generated while the signature verification algorithm must be random in nature. The algorithm used to generate the random number is not cryptographically secure protection against an attacker can utilize this vulnerability i.e. it must be unpredictable so as to the probability of given value being selected should be very small. Since an upcoming scope of our proposed work cryptographically secure random number must be included at the same time as generating private keys.

The space complexity is analyzed between private key length which is in bits and run time memory consumed by system. Space complexity is usually communicated as an order of magnitude, for example O(N^2) , other than the size of the issue (n) get twice subsequently it four times enhanced as working storage capability will be needed. At this point, RSA needs the storage requirements in bytes to perform with an elliptic curve cryptosystem and a 1024-bit modulus over GF(p) wherein p is 160 bits in length while making a rough comparison between the beneath four systems. Table 4 and Figure 4 have shown Space complexity optimizations in an effective manner.

Table 4: Space Complexity (Run Time Memory) 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	348040	341795	335137	327152
2048:224:160:233	348608	341808	334668	325510
3078:256:192:283	349465	342053	334401	324823
7680:384:224:409	358454	350500	342513	332525
15360:521:256:571	368845	360743	352285	342162

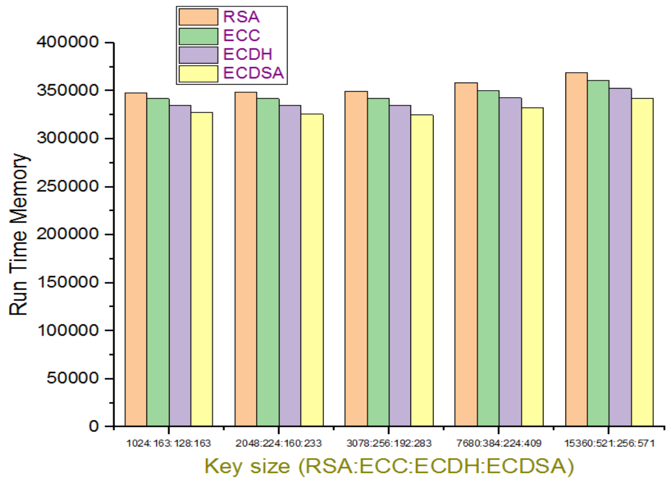


Figure4: Space Complexity (Run Time Memory) Comparison of RSA, ECC, ECDH and ECDSA

IV.CONCLUSION

An encryption and decryption algorithm plays a vital role within data security on the cloud. Different encryption algorithms have been offered to create cloud data secure, vulnerable and provided concern to security challenges and risks. From this paper, the comparisons between **ECC and RSA** based algorithms to find out the best security algorithm that is performed in cloud computing for making cloud data secure and not to be hacked by unauthorized entry or attackers. In this paper for the experimental results exposed **ECDSA** is better performance for the remaining algorithms like **ECC, ECDH, and RSA** according to the space complexity**.** The upcoming scope of this work is to evaluate or find out an efficient proposed algorithm to make the data secure than **ECC and RSA** based algorithms.

V.REFERENCES

[1]. W.Stallings; “Cryptography and Network Security” 2nd Edition, Prentice Hall, 1999
[2]. Bruce Schneir: Applied Cryptography, 2nd edition, John Wiley & Sons, 1996
[3]. Abrams, M., and Podell, H. “Cryptography” Potentials, IEEE Page No 36-38. Issue: 1, Volume: 20, Feb-Mar, 2001
[4]. Eskiciogiu, A. Litwin,L “ Cryptography and Network Security” LOS Alamitos,CA: IEEE computer society press,1987
[5]. Garfinkel, S.L; “Public Key Cryptography”, Computer, IEEE, Volume: 29, Issue: 6, June 1996.
[6]. R. Mayank Kumar, "Developing a QoS Aware Framework for LTE, QoS Control and QoS Aware Scheduler", Aricent White Paper, July 2012.
[7]. Z. Zhu, G. Cao, R. Keralapura and A. Nucci, "Characterizing Data Services in a 3G Network: Usage, Mobility and Access Issues",IEEE International Conference on Communications,pp.1-6, June 2011.

[8]. A. Larmo,"Usability of Instant Messaging over WCDMA", M.S.Thesis, Helsinki University of Technology,Finland,2005.
[9]. Periyanatchi S, Chitra.K. [2015] Analysis on Data Security in Cloud Computing-A Survey. International Conference on Computing and Intelligence Systems 04:1281 – 1284.
[10]. CharanjeetKaur et al. [2015] Data Security Algorithms In Cloud Computing: A Review. International Journal For Technological Research In Engineering 2:372– 375.
[11]. 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.
[12]. 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.
[13]. 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.
[14]. Rashmi S et al. [2015] Architecture for Data Security In Multi-cloud Using AES-256 Encryption Algorithm. International Journal on Recent and Innovation Trends in Computing and Communication 157-161.
[15]. 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)
[16]. 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
[17]. S. Maria Celestin Vigila , K. Muneeswaran "Implementation of Text based Cryptosystem using Elliptic Curve Cryptography", IEEE Sep-2009, pp. 82-85.
[18]. Harsandeep Brar , Rajpreet Kaur, "Design and Implementation of Block Method for Computing NAF" IJCA, Volume 20– No.1, April 2011, pp. 37-41.
[19]. Abhuday Tripathi, and Parul Yadav, Enhancing Security of Cloud Computing using Elliptic Curve Cryptography, International Journal of Computer Applications, 57(1), 2012, 0975-8887.
[20]. Nilesh N. Kumbhar, Virendrasingh V. Chaudhari, and Mohit A.Badhe, The Comprehensive Approach for Data Security in Cloud Computing: A Survey, International Journal of Computer Applications, 39(18), 2012, 0975-8887.
[21]. N. Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, 1987.
[22]. 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, 27 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.