SPACE COMPLEXITY RESEARCH OF CLOUD DATA SECURITY: ELLIPTICAL CURVE AND POLYNOMIAL CRYPTOGRAPHY

[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: Encryption has come up as a solution and different encryption algorithms play an important role in data security on cloud. Encryption algorithms is used to ensure the security of data in cloud computing. Due to some limitations of existing algorithms, there is need for more efficient methods in implementation for public key cryptosystems. Elliptic Curve Cryptography (ECC) is based on elliptic curves defined over a finite field. Elliptic Curve Cryptography has many features that distinguish it from other cryptosystems, one of which is that it is still relatively new cryptosystem. As such, many improvements in performance have been discovered during the last few years for Galois Field operations both in Polynomial Basis and in Normal Basis. . However, there is still some confusion to the relative performance of these new algorithms and very little examples of practical implementations of these new algorithms. Efficient implementations of the basic arithmetic operations in finite fields GF(2^m) are desired for the applications of cryptography and coding theory. The elements in GF(2^m) can be represented in various bases. The choice of basis used to represent field elements has a significant impact on the performance of the field arithmetic. The multiplication methods that use polynomial basis representations are very efficient in comparison to the best methods for multiplication using the other basis representations. ***This paper focuses on user confidentiality protection in cloud computing using enhanced elliptic curve cryptography (ECC)*** algorithm over Galois Field GF(2^m). The Strength of the ***proposed ECPC*** algorithm depends on the complexity of computing discrete logarithm in a large prime modulus, and the Galois Field allows mathematical operations to mix up data easily and effectively. The methodology used involves encrypting and decrypting data to ensure user confidentiality protection and security in the cloud. ***Results show that the performance of ECPC*** over Galois Field, in two area of evaluation, is better than the other algorithm which is used for comparison purpose.

Keywords: Cloud Computing, Data Security, ECC, ECDH, ECDSA and ECPC.

I. INTRODUCTION

Security in cloud computing involves concepts such as network security, equipment and control strategies deployed to protect data, applications and infrastructure associated with cloud computing [1]. An important aspect of cloud is the notion of interconnection with various materials which makes it difficult and necessary securing these environments. Security issues in a cloud platform can lead to economic loss, also a bad reputation if the platform is oriented large public and are the cause behind the massive adoption of this new solution. The data stored in the cloud for customers represents vital information. This is why the infringement of such data by an unauthorized third party is unacceptable. There are two ways to attack data in Cloud [2]. One is outsider attack and the other is insider attack. The insider is an administrator who can have the possibility to hack the user's data. The insider attack is very difficult to be identified. So the users should be very careful while storing their data in cloud storage. Hence, the need to think of methods that impede the use of data even though the data is accessed by the third party, he shouldn't get the actual data. So, all the data must be encrypted before it is transmitted to the cloud storage [3]. Security allows the confidentiality, integrity, authenticity and availability of information. The development of technologies and their standardization makes available a set of algorithms and protocols for responding to these issues [4]. Many encryption algorithms have been developed and implemented in order to provide more secured data transmission process in cloud computing environment, such as, DES, AES, RC4, Blowfish, and 3DES for symmetric category and RSA, DH for asymmetric category [5, 6].

This paper focuses on user confidentiality protection in cloud computing using enhanced elliptic curve cryptography (ECC) algorithm over Galois Field GF(2^m) [7]. The Strength of the proposed ECC algorithm depends on the complexity of computing discrete logarithm in a large prime modulus, and the Galois Field allows mathematical operations to mix up data easily and effectively. Elliptic Curve Cryptography Algorithm provides secure message integrity and message authentication, along with non-repudiation of message and data confidentiality. Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys [8]. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers. Because ECC helps to establish equivalent security with lower computing power and battery resource usage, it is becoming widely used for cloud applications.

Efficient finite field arithmetic is essential for fast implementation of Elliptic Curve Cryptography (ECC) in software environments. Finite field squaring is an important arithmetic operation in the binary finite field GF(2^m). Squaring is required for many cryptographic techniques based on the Discrete Logarithm Problem (DLP) in the multiplicative group of a finite field or additive group of points on an Elliptic Curve defined over a finite field [9]. ***In this paper we present a new method for performing binary finite field squaring in Polynomial Basis.***

Elliptic Curve

Elliptic curve [10] has a unique property that makes it suitable for use in cryptography i.e. it's ability to take any two points on a specific curve, add them together and get a third point on the same curve. The main operation involved in ECC is point multiplication, i.e. multiplication of a scalar K with any point P on the curve to obtain another point Q on

the same curve. An elliptic curve is defined by an equation, is of two variables, with coefficients. For the purpose of cryptography, the variable and coefficients are limited to a special kind of set called a **FINITE FIELD**. The general equation for an elliptic curve is:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d and e are real numbers and x and y also take their values from real number. A simplified elliptic curve equation is given as:

$$y^2 – x^8 + dx + e$$

In **Elliptical Curve Cryptography**, the Elliptic Curve is used to define the members of the set over which the group is calculated i.e. an operation on any two elements of the set will give a result that is the member of the same set as well as operations between them which defines how math work in the group. In real time situation the ECC is implemented over a finite prime field and in hardware, where binary number are used, the field is GF (2^m). The GF is a finite field namely Galois Field. The field of finite primes provides the ECC that allows encipher to encrypt the data very easily but for the cryptologist the process of beginning to attack the encrypted message is very difficult. The GF (p) is the field of integers module p, and consists of all the integers from 0 to p-1 in case of square graph, p*p in size, where p is a very large prime number. To implement the ECC in software to handle prime numbers as well as other number, ECC allows the development of potable chips that can be deployed over the mobile devices and provide a suitable and processor friendly encryption . Like every other design consists of ECC that is implicated on hardware over binary finite fields, point adding and doubling on elliptic curve and scalar multiplication.

1.1 Polynomial Basis Representation

Galois fields are fields with a finite *field order q* which is also the number of elements in the field [11]. A Galois field of order *q* is here denoted as *GF*(*q*). The order *q* of the field is always a prime *p* or a power of a prime *pm* [12]. Galois fields are commonly used for cryptography purposes but there are many applications using Galois fields also e.g. in the study of error-correcting codes. Field orders (field sizes) used for cryptography are usually huge (e.g. $m > 150$) in order to make crypto analysis harder. The security offered by the cryptosystem usually increases exponentially when *m* becomes larger. For cryptography applications *m* in *GF*(2^m) should be a prime, in order to avoid a crypto analysis attack. Galois fields with a polynomial basis are most commonly used in elliptic curve cryptography and therefore they are presented here in detail. The other commonly used basis is called optimal normal basis (ONB), Polynomial basis has proven to be faster and easier to implement than optimal normal basis and it is therefore usually preferred to ONB. Galois field with a polynomial basis is generated with an *irreducible polynomial* over *GF* (2^N).

For the polynomial basis representation, each element of the field represents a polynomial, f(x) of the form:

$$f(x) = a_0 + a_1x + a_2 x^2 + a_3x^3 + a_ix^i$$

All coefficients, a_i, of the polynomial are either zero or one. Therefore each operation, such as addition, subtraction, and multiplication are defined using polynomial arithmetic with the coefficients reduced modulo 2. For example, the bit sequence 01100101 would represent the polynomial:

$$x^6 + x^5 + x^2 + 1$$

Since operations such as multiplication and squaring take inputs of size m bits and result in values 2m-1 bits, there must be a method for reducing the result of these operations into elements of the Galois field. As a result, these values are reduced by a reduction polynomial of order m for an m bit field. Therefore taking the reduction polynomial raised to any power of x and adding it to a field element will result in a value congruent to the original field element modulo the reduction polynomial.

Field operations [13]: the following arithmetic operations are defined on the elements of GF(2^m) when using a polynomial basis representation with reduction polynomial *p(x)*:

Addition Operation

It can be done only using one n-bit XOR operation (equal to bit wise addition module 2). The sum of two elements A, B ∈ GF(2^m) is given by below equation.

$$C(x) = A(x)\ Xor\ B(x) = \sum_{i=0}^{m-1} (a_i\ Xor\ b_i)x_i$$

Square Operation

The binary representation of element's square is done by inserting a 0 bit between consecutive bits of the binary representation. The square of A ∈ GF(2^m) is given by below equation.

$$A^2(x) = \sum_{i=0}^{m-1} a_i x^{2i}$$

Multiplication Operation

Assume we have two elements A(x), B(x) belongs to binary field GF(2^m) with irreducible polynomial P(x). Field multiplication done by two steps [14]:

1. Polynomial multiplication of A(x) and B(x)

$$C'(x) = A(x).\ B(x)$$

2. Reduction using irreducible polynomial p(x)

$$C(x) = C'(x) \bmod P(x)$$

Reduction Operation

Multiplication and square operations need as mention above a reduction process which is the process to reduce the order of resulting values from larger than m to less or equal to m.

$$C(x) = C'(x) \bmod P(x)$$

Inversion Operation

Inversion is the most time-consuming process when computing the scalar multiplication in elliptic curve cryptography using Montgomery method. Inversion in binary finite field is the process of getting a^{-1} for a nonzero element a ϵ GF(2^m) such that:

$$(A.\ A^{-1}) = 1 \bmod f(x)$$

II.SECURITY ALGORTIHMS

A. Elliptic Curve Cryptography (ECC)

Elliptical curve cryptography (ECC) is one of the public key encryption techniques that generate best cryptographic keys according to the elliptic curve theory. It creates smaller keys within a short period. Rather than using large prime numbers for key generation, ECC uses the properties of elliptic curves to generate keys. Elliptic curve is a nonsingular cubic curve with two variables in a certain field and an infinite rational point [15, 16]. Each user generates a public- private key pair, where the public key is applied for encryption and signature verification and the private key is applied for decryption and signature generation. The high level of security can be achieved in ECC using a 164 bit key, where the traditional techniques need 1024 bit key. Data security using ECC algorithm

Key generation

- A selects random integer dA, which is A's private key

- A generates a public key PA = dA * B
- B selects a private key dB and generates a public key PB = dB * B
- A generates the security key Key = dA * PB
- B generates the security key Key = dB * PA

Signature Generation

- For signing a message m by sender of cloud A, using A's private key dA
- Calculate e = HASH(m), where HASH is a cryptographic hash function, such as SHA-1
- Select a random integer k from [1, n-1]
- Calculate r = x1 (mod n), where (x1, y1) = k * B. If r = 0, go to step III
- Calculate s = k^{-1}(e + dA * r) (mod n). If s = 0, go to step III
- The signature is the pair (r, s)
- Finally, send signature (r, s) to B

Encryption algorithm

Assume A sends an encrypted message to B

- A takes plaintext message m, and encodes it onto a point, pm,
- from the elliptic group
- A chooses another random integer, k from interval [1, p-1]
- The cipher text is a pair of points pc = [(kB), (pm + k * PB)]
- Send cipher text pc to B

Decryption algorithm

B will decrypt cipher text pc

- B computes the product of the first point from pc and its private key dB, which is kB * dB
- B takes this product and subtracts it from the second point from pc, (pm + k * PB) – kB * dB, since PB = dB * B, so the difference is pm
- Finally, B decodes pm to get the message m

Signature Verification

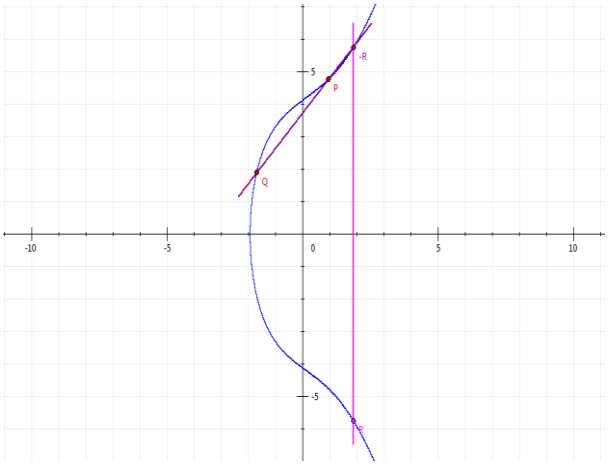
If B wants to authenticate A's signature, B must have A's public key pA

- Verify that r and s are integers in [1, n-1]
- Calculate e = HASH(m), where HASH is the same function used in the signature generation
- Calculate w = (s – 1) % n
- Calculate u1 = e*w % n and u2 = r*w % n
- Calculate (x1,y1) = u1 * B + u2 * PA
- The signature is valid if x1 = r % n, otherwise invalid

EXAMPLE

1. Curve Size: Small, Curve Type: Real number, Curve attributes: a=5, b=17, Curve: $y^2 = x^3 + 5x + 17$, Point P = (0.97|4.77), Point Q = (-1.7|1.89), Point R = P + Q = (1.88|-5.75)

Encryption Chart:



2. Curve Size: Large, Curve Type: F(p), Select curve attributes: ANSI X9.62,Curve: prime192v1, Radix: 16 hexadecimal, Curve attributes: $y^2 = x^3 + 5x + 17$, where
a = fffffffffffffffffffffffffffffffefffffffffffffc
b = 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
p = fffffffffffffffffffffffffffffffefffffffffffffff
Base point G: Point P
x = f2cf5cf93ee95d8748114264ae71b15ca8160d8b777a2e23
y= 8fc23224426e91604820b16640a917d0ca5dfd6ad77abc28
Base point G: point Q
x = f6cc36cbfedacbbb246b8e964095a35f95649b5fd52a6d4b
y=997afb853b3d846f0382fd51b96578e52f625f118e3cd4b7
Point R : R = P + Q
x= 28f240764a3a9bd72bc28dbf29be9174d5952671163c2e5a
y= dab81b6b5b17e7e3d96c23f3831ab1f88465c60cd1ba7c75d

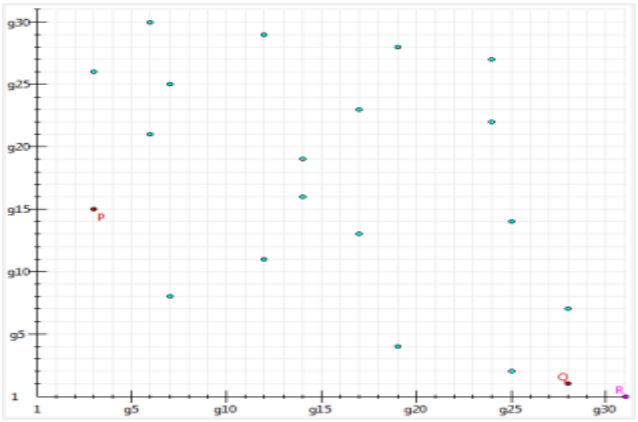
3. Curve Size: Large, Curve Type: F(2^m), Select curve attributes : ANSI X9.62, Curve: c2pnb163v1, Radix : 16 hexadecimal
a = 72546b5435234a422e0789675f432c89435de5242
b = c9517d06d5240d3cff38c74b20b6cd4d6f9dd4d9
m = 163

Base Point P:
x = 00000002 72d6f150 ded9a40e e782567b 93a50953 ea4bf931
y= 00000004 c7884023 97d60585 f8ac1958 14c2120b 3a75ffb3

Base point Q:
X=00000003 a8299d26 75107724 2bcb451a b9f01903 dd1a1c24
Y= 00000003 a1e44106 4c3aa00e 572c7132 e2323827 9eeac2b2
Point R : R = P + Q
X= 00000000 99f7d988 80557118 7fb939c7 009beb7d 57405f8e
Y= 00000004 d5e68ee4 e9d6f508 774a7d6a f83bdc3c e78dfc47

4. Curve Size: Small, Curve Type: F(2^m), curve attributes : m=5, f = x^5+x+1,a=1,b=1,Curve: $y^2 + xy = x^3 + x^2 + 1$, Point P = (g3|g15), Point Q = (g28|g1), Point R = P + Q = (0|1)

Decryption Chart:



B. ECDSA (Elliptic Curve Digital Signature Algorithm)

Signature algorithm is used for authenticating a device or a message sent by the device. For example consider two devices A and B. To authenticate a message sent by A, the device A signs the message using its private key. The device A sends the message and the signature to the device B. This signature can be verified only by using the public key of device A. Since the device B knows A"s public key, it can verify whether the message is indeed send by A or not. ECDSA is a variant of the Digital Signature Algorithm (DSA) that operates on elliptic curve groups. For sending a signed message from A to B, both have to agree up on Elliptic Curve domain parameters. The domain parameters are defined in above section. Sender „A"
have a key pair consisting of a private key dA (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key QA = dA * G (G is the generator point, an elliptic curve domain parameter) [17]. An overview of ECDSA process is defined below:

Signature Generation

For signing a message m by sender A, using A"s private key dA

- Calculate e = HASH (m), where HASH is a cryptographic hash function, such as SHA-1
- Select a random integer k from [1,n − 1]
- Calculate r = x1 (mod n), where (x1, y1) = k *
- If r = 0, go to step 2
- Calculate s = k − 1(e + dAr)(mod n). If s = 0, go to step 2
- The signature is the pair (r, s)

Signature Verification

For B to authenticate A's signature, B must have A"s public key QA

- Verify that r and s are integers in [1,n − 1]. If not, the signature is invalid
- Calculate e = HASH (m), where HASH is the same function used in the signature generation
- Calculate w = s −1 (mod n)
- Calculate u1 = ew (mod n) and u2 = rw (mod n)
- Calculate (x1, y1) = u1G + u2QA
- The signature is valid if x1 = r(mod n), invalid otherwise

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: 349 bytes
Bit length of c + bit length of d = 473 bits
Message = "Elliptic Curve Cryptography Algorithm provides secure message integrity and message authentication, along with non-repudiation of message and data confidentiality. Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys"
Encrypted Data:
45 6C 6C 69 70 74 69 63 20 43 75 72 76 65 20 43 72 79 70 74 6F 67 72 61 70 68 79 20 41 6C 67 6F 72 69 74 68 6D 20 70 72 6F 76 69 64 65 73 20 73 65 63 75 72 65 20 6D 65 73 73 61 67 65 20 69 6E 74 65 67 72 69 74 79 20 61 6E 64 20 6D 65 73 73 61 67 65 20 61 75 74 68 65 6E 74 69 63 61 74 69 6F 6E 2C 20 61 6C 6F 6E 67 20 77 69 74 68 20 6E 6F 6E 2D 72 65 70 75 64 69 61 74 69 6F 6E 20 6F 66 20 6D 65 73 73 61 67 65 20 61 6E 64 20 64 61 74 61 20 63 6F 6E 66 69 64 65 6E 74 69 61 6C 69 74 79 2E 20 45 6C 6C 69 70 74 69 63 61 6C 20 63 75 72 76 65 20 63 72 79 70 74 6F 67 72 61 70 68 79 20 28 45 43 43 29 20 69 73 20 61 20 70 75 62 6C 69 63 20 6B 65 79 20 65 6E 63 72 79 70 74 69 6F 6E 20 74 65 63 68 6E 69 71 75 65 20 62 61 73 65 64 20 6F 6E 20 65 6C 6C 69 70 74 69 63 20 63 75 72 76 65 20 74 68 65 6F 72 79 20 74 68 61 74 20 63 61 6E 20 62 65 20 75 73 65 64 20 74 6F 20 63 72 65 61 74 65 20 66 61 73 74 65 72 2C 20 73 6D 61 6C 6C 65 72 2C 20 61 6E 64 20 6D 6F 72 65 20 65 66 66 69 63 69 65 6E 74 20 63 72 79 70 74 6F 67 72 61 70 68 69 63 20 6B 65 79 73
Elliptic curve E described through the curve equation: y^2 = x^3 + ax + b (mod p) :
a =
883423532389192164791648750360308885314476597252960362792450860609699836
b =
738525217406992417348596088038781724164860971797098971891240423363193866
Private key = 1555396496
Public key W=(Wx,Wy) (W is a point on the elliptic curve) of the signature originator:
W =
498983585649476684779866001437955898822619169124900114917067219042145728
Wx =
257713192384992372601894948053528538433271784776806362284391223808583 58
Wy =
590689380461104336768855452018992077095972376197849521427225501160565422
Calculate a 'hash value' f (message representative) from message M, using the chosen hash function SHA-1.
f =
688647667391257344464700168711587982973589536722

- 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 =
589075071874896415819042058021272878589931382652794723051500724702204335
Gy =
262156637887296030273498439906415542995025924302190305847173246876407 26
r =
883423532389192164791648750360308884807550341691627752275345424702807307
The secret key s is the solution of the EC discrete log problem W=x*G(x unknown)

S=
98479831873905327580543432025919982630968769668634286404736321250868508
Signature:
Convert the group element Vx (x co-ordinates of point V on elliptic curve) to the number i:
i =
25771319238499237260189494805352853843327178477680636228439122380858358
Calculate the number c = i mod r (c not equal to 0):
c =
25771319238499237260189494805352853843327178477680636228439122380858358
Calculate the number d = u^(-1)*(f + s*c) mod r (d not equal to 0):
d =
85761949244041540449303072194687118801333990876964550594656622206742286 6
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 =
30142585998894193338240043683498175662191881525159628439407884297263008
Calculate the number h1 = f*h mod r:
h1 =
10052628669261926303745946084410193019228145856322692122163201254562975 1
Calculate the elliptic curve point P = h1 G + h2 W
Calculate the number h2 = c*h mod r:
h2 =
71618678484096067824334995370229563077640281458090090602336302203984031 70 (If P = (Px, Py) = (inf, inf) then the signature is invalid):
Px =
25771319238499237260189494805352853843327178477680636228439122380858358 Py =
59068938046110433676885545201899207709597237619784952142722550116056542 2
Convert the group element Px (x co-ordinates of point P on elliptic curve) to the number i:
i =
25771319238499237260189494805352853843327178477680636228439122380858358
Calculate the number c' = i mod r:
c' =
25771319238499237260189494805352853843327178477680636228439122380858358
If c' = c then the signature is correct; otherwise the signature is invalid:

C. ECDH (Elliptic Curve Diffie-Hellman Algorithm)

ECDH [18] is a key agreement protocol that allows two parties to establish a shared secret key that can be used for private key algorithms. Both parties exchange some public information to each other. Using this public data and their own private data these parties calculates the shared secret. Any third party, who doesn‟t have access to the private details of each device, will not be able to calculate the shared secret from the available public information. An overview of ECDH process is defined below. For generating a shared secret between A and B using ECDH, both have to agree up on Elliptic Curve domain parameters. The domain parameters are defined in above section. Both end have a key pair consisting of a private key d (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key = d * G (G is the generator point, an elliptic curve domain parameter) [19].

Algorithm: Elliptical Curve Diffie-Hellman

- Alice and Bob agree on the elliptic curve E and base point $G(x_1,y_1)$
- Alice generates a random integer a $\in \{1, \ldots\ldots\ldots, n-1\}$ where n is the order of the group and number a is called private key of the Alice.
- Alice sends to Bob her public key $Q_a=aG=a(x_1,y_1)=(x_a,y_a)$
- Bob generates the random integer b $\in \{1, \ldots\ldots\ldots, n-1\}$ and b number is called private key of the Bob
- Bob sends to Alice his public key . $Q_a=bG=B(x_1,y_1)=(x_b,y_b)$
- Alice can then compute $(x_k,y_k)=aQ_b=a(bG)=abG$.
- Likewise, Bob can compute $(x_k,y_k)=bQ_a=b(aG)=abG$.
- The shared session key is x_k which is the x – coordinate of the point.

EXAMPLE

Step 1: Set public parameters
Curve type: F (p), Curve Size: Small, Domain parameters: a=1,b=1,p=23, generator G=(5,4)
Step 2: Choose Secrets

Alice= 10

Bob=9

Step 3: Generate shared keys

Secret key (d): Q=d*G ,

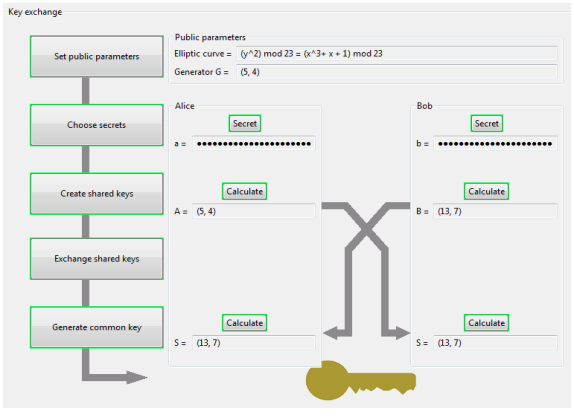
Alice=(5,4)

Bob=(13,7)

Step 4: Exchange shared keys

Step 5: Generate common key

Key = sA*QB and key=sB*QA
S= (13,7)
Exchange shared keys:



D. Proposed ECPC (Elliptical Curve and Polynomial Cryptography)

Polynomial basis multiplication is based on two main arithmetic operations over the binary polynomials: polynomial multiplication and reduction modulo an irreducible polynomial. In this work, the so-called Mastrovito matrix is constructed from the coefficients of the first multiplicand and the irreducible polynomial defining the field. Then, the polynomial multiplication and modulo reduction steps are performed together using matrix. Irreducible polynomials with special structures and low hamming weights have been used in many papers to design efficient finite field multipliers.

In this section, illustrate the representation and multiplication of GF(2^m) elements in the polynomial basis.

The finite field GF(2^m) is an extension field of GF(2) and constitutes a dimension m vector space over it. The finite field GF(2) has only the elements 0 and 1. In this binary field, the addition and the subtraction are defined as XOR operation while the multiplication is defined as AND operation.

Step 1:

Let x ∈ GF(2^m) and be a root of the degree m irreducible polynomial over GF(2)

$$w(x) = x^m + w_{m-1}x^{m-1} + \cdots w_1 x + w_0 = 0$$

Then, the following set constitutes the polynomial basis in GF(2^m):

$$\{1, x, \ldots\ldots\ldots, x^{m-1}\}$$

With polynomial basis, GF(2^m) elements can be represented as degree m-1 polynomials as follows:

$$GF(2^m) = \{a(x)|a(x) = a_{m-1}x^{m-1} + \cdots . + a_1 x + a_0, \quad a_i \in GF(2)\},$$

Where the coefficients a_i are the polynomial basis coordinates in GF(2). When the elements of GF(2^m) are represented as polynomials over GF(2), their addition and subtraction are equivalent to the coefficient-wise XOR, denoted dy "+" in this paper. Also, because of the above equation, all arithmetic operations in GF(2^m) are performed modulo the irreducible polynomial w(x) chosen to construct the field. Let a(x) and b(x) be two field elements and c(x) be their product. Then,

$$c(x) = a(x)b(x) mod\ w(x).$$

Thus polynomial basis multiplication has two steps: polynomial multiplication and reduction modulo an irreducible polynomial.

Step 2: Polynomial multiplication

Let d(x)=a(x)b(x) be the product of the polynomials representing the field elements. D(x) is the degree 2m-2 polynomial

$$d(x) = a(x)b(x) = \left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{i=0}^{m-1} b_j x^j\right) = \sum_{k=0}^{2m-2} d_k x^k$$

Where

$$d_k = \sum_{i+j=k} a_i b_j, \quad 0 \le i, j, \le m-1, 0 \le k \le 2m-2.$$

Step 3: Modular Reduction

In the modular reduction c(x)=d(x) mod w(x), the degree 2m-2 polynomial d(x) is reduced by the degree m irreducible polynomial w(x) iteratively. The partial remainder after each reduction can be computed by the following iteration:

$$d^{(2m-2)}(x) = d(x), d^{(k-1)}(x) = d^{(k)}(x) + w(x)d_k^{(k)}x^{k-m}, m \le k \le 2m-2,$$

Here, $d^{(k)}(x)$ is a partial remainder of degree k and $d^{m-1}(x) = c(x)$. the iteration in above equation reduces $d^{(k)}(x)$ from degree k to k-1, since adding (coefficientwise XORing) $d^{(k)}(x)$ with polynomial

$$w(x)d_k^{(k)}x^{k-m} = \left(x^m + \sum_{i=0}^{m-1} w_i x^i\right) d_k^{(k)}x^{k-m} = d_k^{(k)}x^k + \sum_{i=0}^{m-1} d_k^{(k)} w_i x^{i+k-m} = d_k^{(k)}x^k + \sum_{i=k-m}^{k-1} d_k^{(k)} w_{i-(k-m)} x^i$$

Cancels its term with the order k.

Step 4:
The choice of the irreducible polynomial w(x) may ease the modular reduction. Sparse irreducible polynomials having fewer nonzero terms are usually preferred for efficiency. A degree m irreducible polynomial over GF(2) which has r nonzero terms are in the form

$$x^m + x^{m_1} + x^{m_2} + \ldots . + x^{m_{r-3}} + x^{m_{r-2}} + 1.$$

Step 5:
Here, r > 1 must be an odd number such as 3 and 5. The sparse polynomials with three or five nonzero terms as shown below are called trinomial and pentanomial respectively:

$$x^m + x^{m_1} + 1, x^m + x^{m_1} + x^{m_2} + x^{m_3} + 1.$$

Step 6:
Equally spaced irreducible polynomials are another choice for efficient modular reduction. An equally spaced polynomial is in the form

$$x^{n_s} + x^{(n-1)_s} + \cdots . . x^s + 1,$$

Where ns=m.

Step 7: The proposed polynomial interpolation method in the elliptic curve ElGamal cryptosystem: Now, we are going to discuss the algorithm of the modified elliptic curve ElGamal cryptosystem. This modified cryptosystem will send the set of encrypted points as
two polynomials which are constructed using Lagrange polynomial interpolation method. The first polynomial will be encrypted as well to ensure the additional steps in this modified algorithm is meaningful for implementation. Here the algorithm:

- Alice chooses her secret key, kA such that$1 \le k_A < n$. The gcd (k_A, n) = 1. She publishes her public key as $k_A P$

- Bob chooses k_B such that$1 \leq k_B < n$. The gcd (k_B, n) = 1. He encrypts each points such that (x_E, y_E) = P_m + k_B (k_AP).
- Bob constructs polynomial A(x) based on the points (1, x_{E1}), (2, x_{E2}), (3, x_{E3}), (I, x_{EI}) where l denotes the number of encrypted points. Another polynomial B(x) is constructed based on the encrypted points (x_{E1}, y_{E1}), (x_{E2}, y_{E2}), (x_{E3}, y_{E3}),…, (x_{EI}, y_{EI}). Both polynomials are constructed using Lagrange polynomial interpolation.
- Bob adds the x-coordinate of k_B (k_AP) to each of the coefficients modulo p of the polynomial A(x) whereas the coefficients of polynomial B(x) remain unchanged. The encrypted polynomial A(x) denoted as A' (x). Bob sends (k_BP, A'(x), B(x)) to Alice.
- Alice decrypts by multiplying her secret key such that k_A(k_BP). Polynomial A(x) is obtained from A'(x) by deducting each coefficients using the x coordinate of k_A(k_BP).
- Alice obtains x-coordinate of encrypted points by substituting x = 1, 2,…,I into A(x). Then x coordinate of encrypted points obtained is substituting into B(x) to get the y-coordinate of encrypted points.
- Alice obtains Pm such that P_m + k_B (k_AP) - k_A (k_BP)

Example

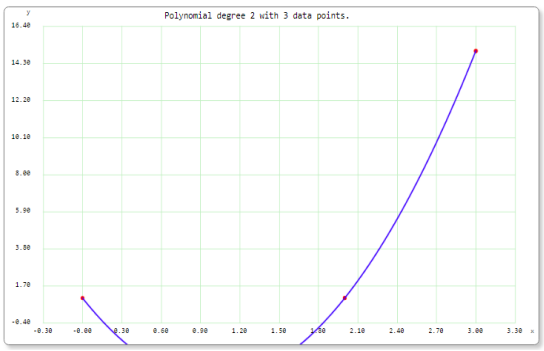
The exponent can be indicated by preceding it by the character E or e, as you can see in the example. Data must consist of two columns, *x* and *y*, to get the polynomial regression $y=a_nx^n+a_{n-1}x^{n-1}+...+a_2x^2+a_1x+a_0$.

Number of Data Points : 3
Polynomial Degree: 2

	x	y	Calculated y	Error
1.	3	15	15	0
2.	2	1	1	0
3.	0	1	1	0

Result: $y = 4.666666667\ x^2 - 9.333333333\ x + 1$
Residual Sum of Squares: rss = 0
Coefficient of Determination: $R^2 = 1$

Chart:



Result

Mode: normal x, y analysis
Polynomial degree 2, 3 x, y data pairs.
Correlation coefficient (r^2) = 1
Standard error = 1.2186183381512667e-14
Coefficient output form: mathematical function:

f(x) = 1.0000000000000111e+000 * x^0
+ -9.3333333333333481e+000 * x^1
+ 4.6666666666666705e+000 * x^2

Table

x,	y	,%
0.00,	1.00,	0.00
0.15,	-0.29,	5.00
0.30,	-1.38,	10.00
0.45,	-2.25,	15.00
0.60,	-2.92,	20.00
0.75,	-3.37,	25.00
0.90,	-3.62,	30.00
1.05,	-3.66,	35.00
1.20,	-3.48,	40.00
1.35,	-3.10,	45.00
1.50,	-2.50,	50.00
1.65,	-1.70,	55.00
1.80,	-0.68,	60.00
1.95,	0.54,	65.00
2.10,	1.98,	70.00
2.25,	3.62,	75.00
2.40,	5.48,	80.00
2.55,	7.54,	85.00
2.70,	9.82,	90.00
2.85,	12.30,	95.00
3.00,	15.00,	100.00

III. RESULTS AND DISCUSSIONS (Based on Space Complexity and Throughput)

In this experimental performance analysis of the given algorithms on the basis of the following parameters on cloud system at different input size. In this section describes the experimental parameters, platforms and key management of experimental algorithms.

Evaluation Parameters Performance of encryption algorithm is evaluated considering the following parameters.

- **Key Generation time:** The Key Generation Time considered the time that a key generation takes to produce a key.
- **Encryption Time:** The encryption time considered the time that an encryption algorithm takes to produces a cipher text from a plain text.
- **Decryption Time:** The decryption time considered the time that a decryption algorithm takes to produces a plain text from a cipher text.

Evaluation Platforms Performance of encryption algorithm is evaluated considering the following system configuration.

- **Software Speciation:** Experimental evaluation on Eclipse Jee Mars with Java Development Kit 8 Update 65, Matlab version 2014, Windows 8.1 Pro 64 bit Operating System.
- **Hardware Speciation:** All the algorithms are tested on Intel Core i5 (2.40 GHz) fourth generation processor with 4GB of RAM with 1 TB-HDD.

Experimental result for encryption algorithm **ECC, ECDH, ECDSA,** and **ECPC** are shown in table-1 which has been implemented several input file sizes: 435 bytes, 869 bytes and 3259 bytes. Key size of each algorithm that is used in this experiment is also mentioned in the below table. All the results are obtained with due care, for ***achieving higher accuracy hundred (150) samples of total execution time*** were taken then an average of hundred samples were taken for the measurement and comparative analysis among algorithms and for the graph plotting as well. Encryption and Decryption time is calculated in millisecond and the input size is taken in kilobytes. All the respective observation readings and graph are shown for all the analyzed algorithms on single system.

Apart from Time complexity, ***space complexity is also an important measure to judge the performance of an***

algorithm. It is the amount of memory which the algorithm needs for performing its computations. A good algorithm keeps the amount of memory as small as possible. The way in which the amount of storage space required by an algorithm varies with the size of the problem it is solving. Space complexity is normally expressed as an order of magnitude, e.g. O(N^2) means that if the size of the problem (n) doubles then four times as much working storage will be needed. We have analyzed the space complexity between private key length which is in bits and run time memory consumed by system. The space complexity of an algorithm is a measure of how much storage is required for a computation. For different input size, there will be different amount of space. Here, we give the storage requirements in bytes of ECC with a 521-bit modulus and an elliptic curve cryptosystem over GF(p) where p is 160 bits in length when making a rough comparison between the below four systems. ***Figure 1 and Table 1 showed Space complexity Comparisons.***

Table 1: Comparison Space Complexity (Run Time Memory)

Key Size ECC:ECDH:ECDSA:ECPC	Space complexity			
521:384:25:163	**ECC**	**ECDH**	**ECDSA**	**ECPC**
521:384:25:163	248040	241795	235137	227152
521:384:25:163	248608	241808	234668	225510
521:384:25:163	249465	242053	234401	224823
521:384:25:163	258454	250500	242513	232525
521:384:25:163	268845	260743	252285	242162

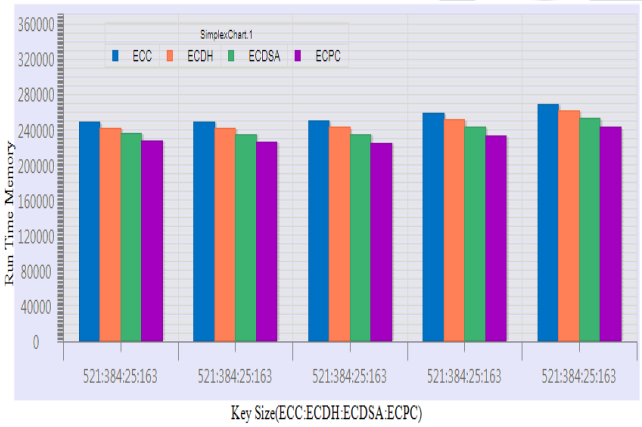


Figure1: Space Complexity (Run Time Memory) Comparison

Throughput

Calculate the throughput of the algorithm by dividing the total data in bytes by encryption time. Higher the throughput higher is the efficiency of the system. Figure 2 and Table 2 given below gives us the comparison between the ECC, ECDH, ECDSA, and ECPC algorithm using throughput. In any cryptographic algorithm, it is essential to understand the size of the input and the size of output as this is one of the important property of an avalanche effect. Figure 2 illustrates the throughput (throughput (Kb/Ms) / time); the corresponding precise measurements are given in figure 2. Our ECPC algorithm differ from others, the proposed algorithm using polynomial based simple and effective operations in elliptic curve. So the ***throughput of the proposed algorithm is higher than others***, the below graph show that proposed algorithm outperforms than others.

Table 2: Comparison using Throughput

Simulated Time/Throughput (Kb/Ms)	ECC	ECDH	ECDSA	ECPC
0	0.000	0.000	0.000	0.000
2	50.000	100.000	150.000	185.000
4	100.000	170.000	190.000	200.000
6	170.000	190.000	220.000	240.000
8	175.000	230.000	250.000	280.000
10	210.000	245.000	265.000	280.000
12	225.000	260.000	280.000	320.000
14	270.000	295.000	320.000	345.000
16	280.000	320.000	340.000	360.000
18	300.000	355.000	375.000	390.000
20	325.000	375.000	410.000	435.000

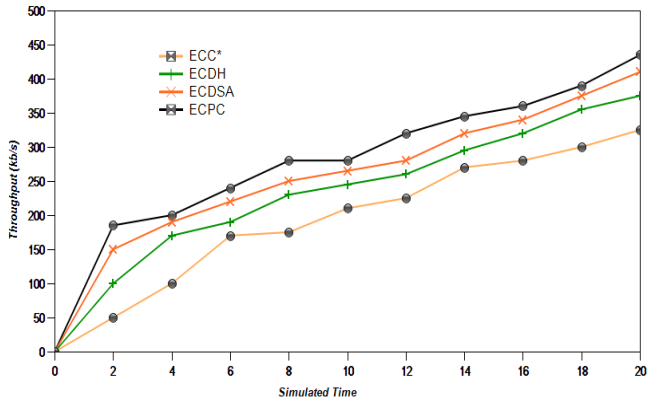


Figure 2: Throughput Comparison

IV. CONCLUSION

Encryption algorithm keeps very important contribution in communication security. This research paper emphasizes on the security of cloud user's information confidentiality protection using enhanced elliptic curve cryptography (ECC) algorithm over Galois Field GF (2^m). The Galois Field allows mathematical operations to mix up data easily and effectively. Our research work showed the performance of widely used encryption techniques like ECC, ECDH, ECDSA and ECPC proposed algorithms. Based on the text files used and the experimental result it has decided that ***ECPC algorithm consumes least run time memory and maximize the throughput*** and ECC(ECC, ECDH, ECDSA) based algorithms ***consume longest run time memory as well as minimize the throughput .***

V. REFERENCES

[1]. Wolf Halton, "Security Solutions for Cloud Computing", July 15, 2010.

[2]. Wolf Halton, Opensource and security on "Security Issues and Solutions in Cloud Computing", July 25, 2010, wolf in cloud computing, Tech Security.

[3]. L. Arockiam, S. Monikandan « Data Security and Privacy in Cloud Storage using Hybrid Symmetric Encryption Algorithm » International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 8, August 2013

[4]. Ponemon Institute and CA "Security of cloud computing Users: A study of Practitioners in the US & Europe". May 12, 2010.

[5]. Ryan K L Ko, Peter Jagadpramana, Miranda Mowbray, Siani Pearson, Markus Kirchberg , Qianhui Liang , Bu Sung Lee, "TrustCloud: A Framework for Accountability and Trust in Cloud Computing" 2011 IEEE World Congress on Services.

[6]. Muhammad Rizwan Asghar, Mihaela Ion, Bruno Crispo, "ESPOON Enforcing Encrypted Security Policies in Outsourced Environment", 2011 Sixth International Conference on Availability, Reliability and Security.

[7]. Xu Huang, Pritam Gajkumar Shah and Dharmendra Sharma, "Multi-Agent System Protecting from Attacking with Elliptic Curve Cryptography," the 2nd International Symposium on Intelligent Decision Technologies, Baltimore, USA, 28-30 July 2010.

[8]. Xu Huang, Pritam Shah, and Dharmendra Sharma, "Minimizing hamming weight based on 1"s complement of binary numbers over GF (2m)," IEEE 12th International Conference on Advanced Communication Technology, Phoenix Park, Korea Feb 7-10, 2010. ISBN 978-89-5519-146-2, pp.1226-1230.

[9]. Xu Huang, Pritam Shah, and Dharmendra Sharma, "Fast Algorithm in ECC for Wireless Sensor Network," The International MultiConference of Engineers and Computer Scientists 2010, Hong Kong, 17-19 March 2010. Proceeding 818-822.

[10]. Pritam Gajkumar Shah, Xu Huang, Dharmendra Sharma, "Analytical study of implementation issues of elliptical curve cryptography for wireless sensor networks," The 3rd International Workshop on RFID & WSN and its Industrial Applications, in conjunction with IEEE AINA 2010, April 20-23, 2010, Perth, Australia.

[11]. Omura, J.K., Massey, J.L.: Computational method and apparatus for finite field arithmetic, United States Patent 4,587,627 (1986)

[12]. Robert, J., McEliece: Finite Fields for Computer Scientists and Engineers. The Kluwer International Series in engineering and computer science. Kluwer Academic Publishers, Dordrecht (1987)

[13]. Karatsuba, A., Ofman, Y.: Multiplication of multidigit numbers on automata. Sov. Transaction Info. Theory 7(7), 595–596 (1963)

[14]. Rodriguez-Henriquez, F., Kog, Q.K.: On Fully Parallel Karatsuba Multipliers for GF (2m).In: International Conference on Computer Science and Technology (CST), pp. 405–410 (2003)

[15]. Setiadi, I., Kistijantoro, A.I. and Miyaji, A. (2015) Elliptic Curve Cryptography: Algorithms and Implementation Analysis over Coordinate Systems. 2015 2nd International Conference on Advanced Informatics: Concepts , Theory and Applications, Chonburi, 19-22 August 2015, 1-6. https://doi.org/10.1109/icaicta.2015.7335349

[16]. J. Krasner "Using Elliptic Curve Cryptography (ECC) for Enhanced Embedded Security –Financial Advantages of ECC over RSA or Diffie-Hellman (DH)" Embedded Market Forecasters American Technology International, Inc. November 2004

[17]. D. Hankerson, A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography"Ch- , Pp 76-78 Springer, 2004.

[18]. Garg, V. and Ri, S.R. (2012) Improved Diffie-Hellman Algorithm for Network Security Enhancement. International Journal of Computer Technology and Applications , 3, 1327-1331.

[19]. Setiadi, I., Kistijantoro, A.I. and Miyaji, A. (2015) Elliptic Curve Cryptography: Algorithms and Implementation Analysis over Coordinate Systems. 2015 2nd International Conference on Advanced Informatics: Concepts , Theory and Applications, Chonburi, 19-22 August 2015, 1-6. https://doi.org/10.1109/icaicta.2015.7335349

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 **18 years** of experience in academic field. He has published **5 books**, more than **35 papers** in International Journals and **30 papers** in National & International Conferences so far. His areas of interest include Computer Networks, Grid Computing, Cloud Computing and Mobile Computing.