Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.8 No.1, 2019, pp.53-61

DOI: https://doi.org/10.51983/ajcst-2019.8.1.2118

Comparative Analysis of Hash Authentication Algorithms and ECC Based Security Algorithms in Cloud Data

S. Hendry Leo Kanickam[1] and L. Jayasimman[2]
[1]Research Scholar, Department of Computer Science, [2]Assistant Professor, Department of Computer Applications
[1&2]Bishop Heber College, Trichy, Tamil Nadu, India
E-Mail: dr.l.jayasimman@gmail.com
(Received 16 January 2019; Accepted 11 February 2019; Available online 17 February 2019)

***Abstract* - Cloud computing is ensuring the security of stored data in cloud computing servers is one of the mainly demanding issues. In Cloud numerous security issues arises such as authentication, integrity and confidentiality. Different encryption techniques attempt to overcome these data security issues to an enormous extent. Hashing algorithm plays an important role in data integrity, message authentication, and digital signature in modern information security. For security purpose using encryption algorithm like ECC (Elliptic Curve Cryptography) and Authentication of data integrity using hashing algorithms like MD5, and SHA-512. This combination method provides data security, authentication and verification for secure cloud computing.**

***Keywords:* Cloud Computing, Hash Algorithm, Elliptic Curve Cryptography, MD5, SHA, ECDH, ECDSA**

I. INTRODUCTION

Cloud computing covers several technologies which includes databases, networks, virtualization, operating systems, load balancing, transaction management, resource scheduling, and concurrency control. It provides potential helps such as cost reductions and developed business outcomes for the business concerns. On the other hand, cloud data security problems for many of these technologies and systems are applicable to cloud computing these risks are required to be carefully considered. In case, these risks will differ based on the sensitivity of the data to be processed or stored, and how the selected Cloud Service Provider has implemented for their specific cloud services[1]. Data is secured only if it fulfills the integrity, confidentiality and availability of data. Therefore, protecting the authenticity, integrity and confidentiality of the data in cloud storage and communications is extremely important. This level of security is obtained by using cryptographic algorithms were developed [2]. Cryptographic algorithms are separated into two forms such as Encryptions (Symmetric and Asymmetric) algorithms and Hashing. The main variation between hashing and encryption of data is that Hashing does not decrypt data like Symmetri3c and Asymmetric encryption performances [3].A wide range of encryption algorithms such as DES, AES, Blowfish and TDES available to make data secure by converting it to unreadable format.

In case, internet attacks are also increased with the hasty improvement of network technology, the existing encryption techniques is not sufficient for information security over internet, therefore the combination of encryption and encoding is broadly being used to attain integrity and confidentiality. There is need of a system that fulfills encrypted data transfer, verification and authentication, therefore maintaining data confidentiality, to obtain rid of the same and to induce trust in the computing. The most important use of hybrid cryptographic algorithm merges with digital signature and encryption algorithm to secure data stored in cloud. Subsequently, the encryption algorithm is used for data confidentiality and digital signature is used for authentication purpose. In this paper, the performance of ECDH, ECDSA and ECC, encryption techniques and SHA-512 encoding techniques are considered to be effectively performed.

II. HASHING ALGORITHM

A cryptographic hash function is efficiently compresses message of any length to certain fixed length (message-digest). This process is irreversible. Hash algorithm comprises Secure Hash Algorithm (SHA), Message Digest Algorithm (MD), RIPE-MD, HAVAL and N-Hash and so on [4].It has three additional security properties such as first and second pre image-resistant and collision-resistant. A function is considered as first pre image-resistant, if having the output value of the function; it is infeasible to discover any input that map to that output. This property is also known as one-way communication as one can easily compute the function in one way but does not return to other direction. In second pre image-resistant, even if given the input value and the corresponding output it is computationally infeasible to discover another distinct input to facilitate hashes to the same output. As a final point, a function is known as collision-resistant, if it is infeasible to discover a pair of different inputs that map to the identical value.

A function satisfying all the properties mentioned above is a powerful and versatile tool and can be used to achieve a variety of security goals. Hash function is used in several fields like digital signature, information authentication, message integrality test, and message originality and one of the very common applications of cryptographic hash functions is mainly used for password protection in access control systems. Hash algorithms are differentiated using

the size of the hash value that they generate as follows. In MD Algorithm Family were using 128 bits, and Secure Hash Algorithms using 160 bits. Ron Rivest was developed the Message Digest techniques for performing digital signature applications. MD2 is cryptographic hashing algorithm used in 8 bits systems. MD4 and MD5 is also cryptographic hashing algorithm used in 32bit systems. MD5 is slower than MD4 but it has more secured compare than MD 4. National Institute for Standards and Technology (NIST) was developed the Secure Hash Algorithm (SHA/SHA-1/SHA-2). SHA-2 is more secured than SHA-1.

A. Message Digest 5 (Md5)

In cryptographic hashing algorithm, MD-5 is used to generate a 128 bits fixed length hash value [5]. MD5 is slower than MD4 but it has more secured compare than MD 4 and using the performance of 32-bit processors. An emphasis on 32-bit processors is completed because, the four word buffers (P, Q, R, S) that are used to compute the message digest are, each, a 32-bit register.

The given below steps explain the overview of MD5 operations [6].

1. Steps Overview

a. Pad message so its length plus 448 is divisible by 512.
b. Appending message is end from a 64-bit value.
c. Starting the Four word using 128-bit and buffer (P,Q,R,S).
d. Execution of text in 16-word using 512-bit blocks: Every chunk & buffer will process 4 rounds and 16 bit operations, in which results to pass input and create a new final buffer value.

Each round is computed by [5].
1. Each round has 16 steps:
2. $R(a,b,c,d,M_i,s,t_i)$: $a = b + ((a+F(b,c,d)+M_i+t_i)<<s)$
3. where F(b,c,d) is a different nonlinear function in each round
4. t_i is a value derived from sine

2. Description of MD5

MD5 is the most significant process as same MD4 that includes the same steps of appending and padding bits.With the support of MD5 buffers consequently processed and grouped in 4 rounds of 16 operations are followed and the message in 16-word blocks consists of 64 operations by the final output. MD5 uses three operations such as Bitwise Boolean operation, Cycle shift operation and Modular Addition operation.MD5 performs in two steps like Compression and Padding that must be performed effectively.

a. Padding
1. In padding operation, input message is broken up into 512 bit blocks with the intention that input length is dividable by 512. Initially, a single bit appends at the end of the message. Afterward a series of 0's are appended with the intention that the length of the padded message is congruent 448 mod 512.
2. The length of the message represents in 64 bit binary string even if the message is too long, greater than 264 then lower 64 bits are used for binary representation.

b. Divide the Message

MD5 processes the input string in 512-bit blocks, partitioned into16 operations 32-bit sub-blocks to form single 128-bit hash value in the output of the algorithm.

c. Initialization of the State Variable

MD5 utilizes 4 state variables that is a 32 bit integer. These four variables are sliced and diced such as P,Q,R,S initializations as follows:

P=0x67452301
Q=0XEFCDAB89
R=0x98BADCFE
S=0x10325476

d. Table

MD5 further utilizes a table K that has 64 elements. Element number i specifies as K_i. The table computes beforehand to get faster the computations. The mathematical sin function computes by the elements:

$$K_i = | \sin(i + 1)| * 2^{32}$$

e. Compression

Four functions are used in this algorithm as follows.

F(X, Y, Z) =(X&Y) | ((~X) &Z)
G(X, Y, Z) =(X&Z) | (Y& (~Z))
H(X, Y, Z) =X^Y^Z
I(X, Y, Z) =Y ^ (X|~ (Z))

At this point &, |,^,~ are bitwise AND, OR, XOR and NOT operator in which each 512 bits is performed. After this step the result which is in the message digest form is stored in the state variable A, B, C, D.

f. Processing the Blocks

The whole contents of the four buffers are mixed with the words of the input (A, B, C and D), using the four functions (F, G, H and I). There are four rounds, each involves 16 basic operations. The four copied into the different variable: *a* gets *A*, *b* obtains *B*, *c* acquires *C*, and *d* obtains *D*. The main loop has four rounds and each series utilizes a different operation 16 times. Each operation performs a nonlinear function on three of *a*, *b*, *c*, and *d*. After that, it adds that result to the right a variable number of bits and adds the result to one of *a*, *b*, *c*, and *d*.

As a final point, the result replaces one of *a*, *b*, *c*, and *d*. One operation is demonstrated in the figure below.

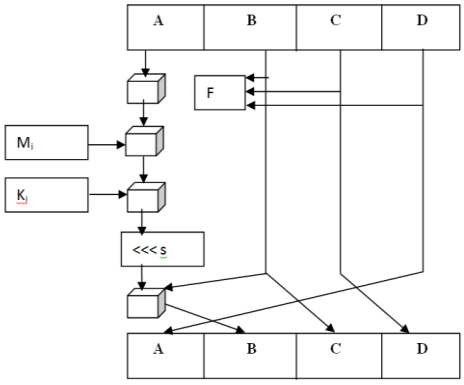

Fig. 1MD5 Processing

In this figure, it illustrates how the auxiliary function F is operated through the four buffers (A, B, C and D), using constant Ki and message word Mi. The item "<<<s" denotes a binary left shift by *s* bits.

g. Hashed output

Next step, all rounds have been performed, the buffers A, B, C and D contain the MD5 digest of the original input. The final output is recognized as a hash value, a fingerprint or a message digest.

B. Secure Hash Algorithm -2 (SHA-2)

SHA is a cryptographic hashing algorithm and the new versions SHA-2 bear the same underlying resemblance of structure, modular arithmetic, and logical binary operations as that of SHA-1 without sharing its weaknesses [7].Each data block has 512 bits with the purpose of indicated as a sequence of 32-bit words with the support of SHA- 256 and SHA-224 bits. Each data block contains 1024 bits as a sequence of 64-bit words with the support of SHA-512 and SHA-384.Each 32-bit words and 64-bit words operated by using SHA-256 and SHA-512.Both SHA-512 and SHA-256 are fresh hash functions that utilize different additive constants and shift amounts and their structures are virtually identical, differing only in number of rounds.

The final version of SHA-512 had been performed from SHA-224/SHA-256. Here is a step summary of SHA-512 processing [8,9].

1. Steps Overview

a. SHA-512 is a variant of SHA-256 which operates on eight 64-bit words. Start hashed function for message. Padded length - multiple of 1024bits long
b. 1024-bit message blocks $M^{(1)}$, $M^{(2)}$, ….., $M^{(N)}$ is parsed
c. and processed one time.
d. Initial hash value $H^{(0)}$, to $H^{(i)} = H^{(i-1)} + C_M^{(i)}(H^{(i-1)})$,
e. Where C - Compression function and. $H^{(N)}$- hash of M.
f. 64 bit word Output are generate from SHA-512's 6 logical function and each function worked as x,y,z with 64 bit words.

2. Narrative of SHA-512

SHA-512 is the overall processing of show by given below figure 2. The different processing steps of SHA-512 presents in detail as given below.

a. Append Padding Bits and Length Value

In this step makes the input message an exact multiple of 1024 bits:

1. The length of the total message to be hashed should be a multiple of 1024 bits.
2. The last 128 bits of what obtains hashed are reserved for the message length value.
3. Original message as an exact multiple of 1024 and need to append another 1024-bit block at the end. In which process to create block for the 128-bit message.
4. The padding initiate 1 bit after needed 0 bits with 128 bit messages $\{M_1, M_2, \ldots, M_N\}$,
5. The length value in the trailing 128 bit positions is an unsigned integer with its most significant byte first.

b. Initialize Hash Buffer with Initialization Vector

1. The hash buffer execute by eight 64-bit registers(a, b, c, d, e, f, g, h)
2. Initialized by the 64 bits of the fractional parts of the square-roots of the first eight primes in registers. These are shown below in hex:
   6a09e667f3bcc908bb67ae8584caa73b3c6ef372fe94f82
   ba54ff53a5f1d36f1510e527fade682d1
   9b05688c2b3e6c1f 1f83d9abfb41bd6b
   5be0cd19137e2179

c. Process each 1024-bit Message Block M_i

1. Must generate what is recognized as a message schedule. The message schedule for SHA-512 contain 80 rounds of 64-bit words labeled $\{W_0, W_1, \ldots, W_{79}\}$. The first sixteen of these, W_0 through W_{15}, are the sixteen 64-bit words in the 1024-bit message block M_i. The rest of the words in the message schedule are obtained by
   $W_i = W_{i-16} + 64\ \sigma_0 (W_{i-15}) + 64\ W_{i-7} + 64\ \sigma_1 (W_{i-2})$
   $\sigma_0(x) = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$
   $\sigma_1(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x)$
   $ROTR^n(x)$ = circular right shift of the 64 bit arg by n bits
   $SHR^n(x)$ = right shift of the 64 bit arg by n bits with padding by zeros on the left
   +64 = addition module 2^{64}
2. Round-based processing of input messages, the i^{th} round is fed the 64-bit message schedule word W_i and a special constant K_i.
3. Message block $M_i$has two inputs: the current contents of the 1024-bit message block and the 512-bit hash buffer

4. The round function consists of a sequence of transpositions and substitutions, all designed to diffuse to the maximum extent possible the content of the input message block. The relationship between the contents of the eight registers of the hash buffer at the input to the i^{th} round and the output from this round is given by

$h = g$
$g = f$
$f = e$
$e = d + 64\ T_1$
$d = c$
$c = b$
$b = a$
$a = T_1 + 64\ T_2$

Where +64 again means modulo 2^{64} addition and where
$T_1 = h + 64\ Ch(e, f, g) + 64 \sum e + 64\ W_i + 64\ K_i$
$T_2 = \sum a + 64\ Maj(a, b, c)$
$Ch(e, f, g) = (e\ AND\ f) \oplus (NOT\ e\ AND\ g)$
$Maj(a, b, c) = (a\ AND\ b) \oplus (a\ AND\ c) \oplus (b\ AND\ c)$
$\sum a = ROTR^{28}(a) \oplus ROTR^{34}(a) \oplus ROTR^{39}(a)$
$\sum e = ROTR^{14}(e) \oplus ROTR^{18}(e) \oplus ROTR^{41}(e)$
$+64$ = addition modulo 2^{64}

5. The final output of the 80^{th} round is added at the initiating of the round-based processing to the content of the hash buffer. Each 64-bit word of the output of the 80^{th} modulo $2^{64}$are performed using addition operation.

d. Final

After that process, all the N message blocks have been processed; the content of the hash buffer is the message digest.

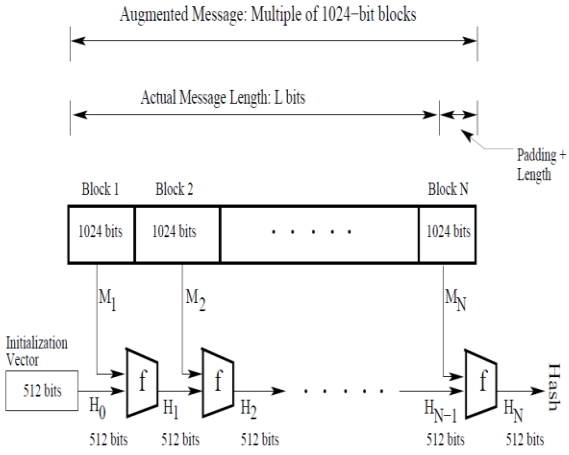

Fig.2Overall processing of the SHA-512 [10].

Example:

Input Type: TEXT
Content: Cloud Computing
Hash List [Hash Name (Length) - Hash Data]
SHA512 (64)
a4199f65e484c1df316f2581513ceae75698b372485
151c006d63fda5412cc7b2b051827759b256949696
69c60ff8811b4e58b67660a183218bd5e1a2dba6fa4
MD5 (16) –
3de3a4a550829dc4376329c29771bf78

TABLE I COMPARISON OF FEATURES OF MD5 AND SHA-512 ALGORITHMS

S. No.	Features	MD5	SHA-512
1	Year	1990	1993
2	Message digest length	128	512
3	Security	Less secure	More secure
4	Speed	Faster	Slower
5	No of steps	64 (4 rounds of 16)	80 (4 rounds of 20)
6	Collision complexity	2^{64}	2^{80}
7	Successful attacks Reported	Yes	Yes

III. RESULT AND DISCUSSION OF MD 5 AND SHA

For example, a system requires to design to obtain as input a plaintext message, to analyze the message digest. For this reason, a high level programming language used, JAVA and implemented algorithm using this language. The Net-beans IDE 8.1 Platform and JAVA language is used to achieve the experimental results. The experiments were performed on Windows 10 Pro with Intel(R) Core(TM) i5-2430M CPU @2,40GHz (4CPUs), ~2.4 GHz architecture, and 8.00 GB RAM. The performance of MD5, SHA-512 algorithms verified on 32-bit processor, the MD5 has better performance as made known in following figures.

A. Timing Analysis

The efficiency of an algorithm is mainly used to evaluate the parameter of time analysis. An algorithm is efficient even if it obtains less time to evaluate the digest. Figure 3 illustrate an experimental result after testing it on 25 sample files of same size for each different file size. Graphical representation of algorithms result are shown figure 3. At this point, color line proves the execution time in seconds of different algorithms for a 5 KB file, 10 KB file and 20KB file. Finally, SHA-512 is compared with MD5 that the performance of SHA-512 was slightly better than that of MD5, but SHA-512 exhibited the lowest run time performance. The SHA-512 tends to demonstrate an exponential-like runtime execution.

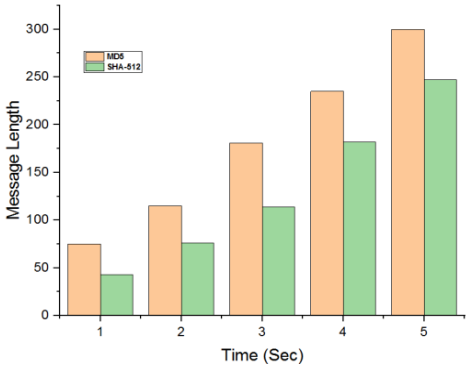

Fig.3 Timing Comparisons between MD5, SHA-512

B. Security Analysis

In securityanalysis,another important aspect is included to evaluate the efficiency of hash algorithm is its security. Security of hash algorithm can be computed with the support of avalanche effect and it has the two similar message having difference of single bit only generates a digest which results different from each other in 50 percent bits. The algorithm nearer to the ideal condition is more secure and the algorithms far away from this condition considered less secure. MD5 algorithm has better security level to compare SHA-512.

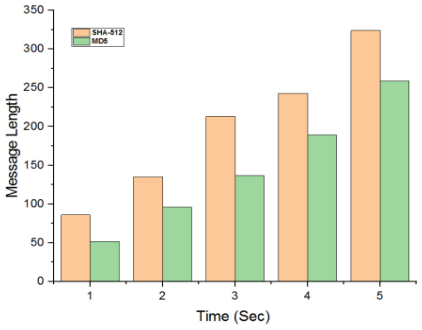

Fig. 4 Security comparison of MD5, SHA-512

IV. ENCRYPTION ALGORITHMS

A. ECC

Elliptic Curve Cryptography is faster than RSA and utilizes smaller keys, however still affords reducing transmission and storage requirements, the same level of security. It can be solved the elliptic curve discrete log problem but it's a much harder issue than factoring integers. ECC presents the security per bit of public key technique [11].

1. Key Generation

In key generation, both public key and private key is an important part to be generated in an efficient manner. The message with receivers receives the public key from the sender. The receiver will decrypt its private key.
To choose a number 'd' within the range of 'n'.
To generate the public key using the following equation
Q = d * P
d = The random number is selected within the range of (1 to n-1). Pis the point on the curve.
'Q' is the public key and 'd' is the private key.

2. Encryption

Let 'm' be the message that we are sending. We have to represent this message on the curve.
Consider 'm' has the point 'M' on the curve 'E'. Randomly select 'k' from [1 - (n-1)].
Two cipher texts will be generated let it be C1 and C2.
C1 = k*P
C2 = M + k*Q
C1 and C2 will be send.

3. Decryption

We have to get back the message 'm' that was send to us,
M = C2 – d * C1
M is the original message that we have send.
How do we get back the message?
M = C2 – d * C1
'M' can be represented as 'C2 – d * C1′
C2 – d * C1 = (M + k * Q) – d * (k * P) (C2 = M + k * Q and C1 = k * P)
= M + k * d * P – d * k *P (canceling out k * d * P)
= M (Original Message)

Example:

1. Curve Size

Small , Curve Type: Real number, Curve attributes: a=2, b=15, Curve: $y^2 = x^3 + 2x + 15$, Point P = (0.75|-4.11),Point Q = (0.72|4.1),Point R = P + Q = (108007.77|2.147483647E7) (See Fig. 5)

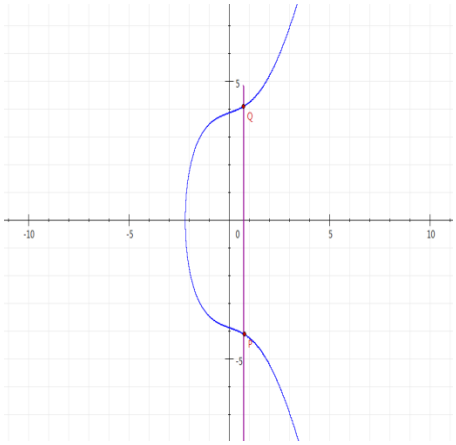

Fig. 5 Curve Points

2. Curve Size

Large, Curve Type: F(p), Select curve attributes: ANSI X9.62,Curve: prime192v1, Radix: 16 hexadecimal, Curve attributes: $y^2 = x^3 + 2x + 15$,(See Figure 6) where,
a = fffffffffffffffffffffffffffffffefffffffffffffffc,
b = 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
p = fffffffffffffffffffffffffffffffefffffffffffffff
order of G : fffffffffffffffffffffff99def836146bc9b1b4d22831
Base point G: Point P
x = 12254520284046a953aad054a56f37f32d3ee74a70cc16c1
y= a74eb0b373c203b72bf01404203f687186574da64fd45df9
Base point G: point Q

x = c3cbb6f435b02845143702e090677a8d48172e31d983293b
y= 17155a5dfda5f0a489d54fe2f3ef2c35d40990b2f9f586b7
Point R : R = P + Q
x = 63c0513338ab30f3b52907f120660bf907c9aca5b9ad2525
y= 89ab4e8537d8b868844a53bbbc420a5aae1a6927d1a4856b

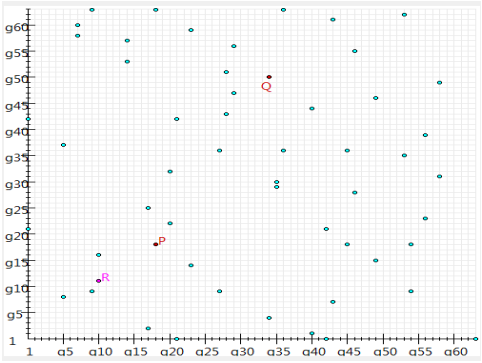

Fig. 6Curve Points of P and Q

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 = 00000003 6aa9c720 620ea74a 1b98fd83 5bf3fa93 0816de24
y= 00000003 d7e0fd08 0e2841a4 872c9e03 52e50799 519c9270
Base point Q:
X= 00000000 ab36ea3a e3b70610 5ff2574f 498a1ab9 82cae478
Y= 00000005 06b2493c ea737aba 87a370ef 716a6f0f f3d2c6b8
Point R : R = P + Q
X= 00000001 83028a9b 677ea42d 19248f24 31c24e60 425fc86e
Y= 00000007 7743fd9d 43819576 eb3b90bc 0869de99 3165af21

4. Curve Size

Small, Curve Type: F(2^m), curve attributes : m=6, f = x^6+x+1,a=1,b=1,Curve: $y^2 + xy = x^3 + x^2 + 1$, Point P = (g18|g18), Point Q = (g34|g50), Point R = P + Q = (g10|g11)

B. ECDHA

ECDH is a key agreement protocol and it provides a shared authentication key between P and Q [12,13]. A key pair contains a private key d and a public key R= d * G (G is the generator point). Let (d_p, R_p) be the private key - public key pair of P and (d_q, R_q) be the private key - public key pair of P and the overview of ECDH is given below.

Both the P and Q have
1. The end P calculates $K = (X_K, Y_K) = d_p * R_q$
2. The end Q calculates $L = (X_L, Y_L) = d_B * R_A$
3. Since $d_pQ_q = d_pd_qG = d_qd_pG = d_qQ_p$. Therefore K = L and hence $X_K = X_L$
4. Therefore the shared secret is X_K

Example:
Step 1: Set public parameters
Curve type: F (p), Curve Size: Small, Domain parameters: a=2,b=15,p=29, generator G= (5,11)
Step 2: Secrets are selected
Alice= 12, Bob=9
Step 3: The shared keys are generated
Secret key (d): Q=d*G , Alice=(5,18), Bob= (11,11)
Step 4: Exchange shared keys
Step 5: Generate common key
Key = sA*QB and key=sB*QA
S= (11,18)

C. ECDSA

In ECDSA (Elliptic Curve Digital Signature Algorithm is an ECC scheme that requires a few modular operations and a hash function. ECDSA is relatedtoElGamal's signature technique other than it uses a slightly different signature verification method that makes verification of signatures faster [14].

The main difference between ECDSA and ElGamal's digital signature system is that in ElGamal's system the verification process requires three multiplications of an integer times a point, whereas in ECDSA only two multiplications of an integer times a point are required. These multiplications are the most expensive parts of these algorithms.

Given below explain the procedure of ECDSA. In ECDSA, the signature generation and verification is similar to DSA, but the key generation is based on ECC algorithm [15,16].

1. Key Pair Generation

In ECDSA, key pair generation is based on the domain parameters. Given the elliptic curve E over Z_p with number of points that is divisible by the large prime n,

a. Choose a point $P(x_p, y_p)$ on the curve and a random integer $d \in [1, n-1]$.
b. Compute $Q(x_q, y_q) = dP$, make sure the point Q is also on the curve.
c. Public key is (E,P, n,Q), and private key is d.

2. Signature Generation

Given a message m to be signed and the private key d,

a. Select a random integer k ∈[1, n − 1].
b. Evaluate $(x_1, y_1) = kP$, convert x_1 into integer and $r = x_1 \bmod n$. (Return to step 1 if
   b) r = 0)
a. Compute $s = k^{-1}(SHA1(m) + dr)$. (Return to step 1 if s = 0)
b. Signature is (r, s) pair.

3. Signature Verification

Given the signature pair (r, s) on message m and public key (E,P, n,Q),

1. Check that integers r, s ∈[1, n − 1].
2. Compute $w = s^{-1} \bmod n$.
3. Compute $u_1 = SHA1(m)w \bmod n$, $u_2 = rw \bmod n$.
4. Calculate $(x_0, y_0) = u_1P + u_2Q$, convert x_0 into integer and $v = x_0 \bmod n$.
5. Compare v and r, accept the signature only if v = r.

Example:
ECDSA Key Generation

Signature originator: hendrihendri
Domain parameters to be used 'EC-prime239v1':
Chosen signature algorithm: ECSP-DSA with hash function SHA-1
Size of message M to be signed: 9 bytes
Bit length of c + bit length of d = 476 bits
Message m = "Cloud Computing"
20 20 43 6C 6F 75 64 20 43 6F 6D 70 75 74 69 6E 67 Cloud Computing
Elliptic curve E described through the curve equation: y^2 = x^3 + ax + b (mod p) :
a =
883423532389192164791648750360308885314476597252960362792450860609699836
b =
738525217406992417348596088038781724164860971797098971891240423363193866
Private key = 1537859914
Public key W=(Wx,Wy) (W is a point on the elliptic curve) of the signature originator:
Wx =
147725686096033178777934248266418837925095559073411824357492706021376809
Wy =
160919462516789063671317533636959973111179312851645365948486100202602809
Calculate a 'hash value' f (message representative) from message M, using the chosen hash function SHA-1.
f =
1458831800945090464484162373837128997320441648540
ECDSA Signature:
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=460341977657942268451985328724025241158573155544832179679276669703259603
Signature:
c =
147725686096033178777934248266418837925095559073411824357492706021376809
d =
595522074955150246352225841149857845213076100858728291094353301485229658
ECDSA Verification:
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 =
712639260515375087861719638729379586887201669491882073893208549199833159
Calculate the number h1 = f*h mod r:
h1 =
548405845492297792579891043651695754594577312396988397614672115226793 39
Calculate the number h2 = c*h mod r:
h2 =
402971820697583772099104763357788971108070894764953834845947167468274080
Calculate the elliptic curve point P = h1 G + h2 W
(If P = (Px, Py) = (inf, inf) then the signature is invalid):
Px =
872657954012521758455980834210393555654374469957427229761109261848254598
Py =
733938488033337736229825503670057677858195852917805158753447088809562956
Convert the group element Px (x co-ordinates of point P on elliptic curve) to the number i:
i =
872657954012521758455980834210393555654374469957427229761109261848254598
Calculate the number c' = i mod r:
c' =
872657954012521758455980834210393555654374469957427229761109261848254598
If c' = c then the signature is correct; otherwise the signature is invalid: Verify results by comparing the two numbers c' and c.

V. RESULT ANALYSIS

Measured performance of ECC, ECDH, and ECDSA operations working under OpenSSL0.9.6b speed program (enhanced to include ECC) on two platforms: (i) Ubuntu, a Linux PDA equipped with a 200MHz Strong ARM processor, and (ii) an UltraTM-80, a Sun server equipped with a 450MHz UltraSPARC II processor. Given below Figures demonstrate the performance, at the protocol level, of ECC, ECDH, ECDSA.

A. Key Generation Time

For performance measure ECC, ECDH and ECDSA implementation is entirely constant time. For ECDH, both ours and OpenSSL's optimized NISTP implementations are constant-time. In Ubuntu, the implementation is *2.33x* faster for ECDSA sign, *1.86x* faster for ECDSA verify, and *1.8x* faster for the ECC key computation.

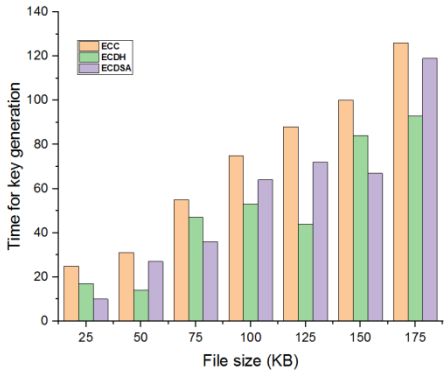

Fig. 7 Key generation time analysis

The Linux performance shows an even larger gap, with the respective speedup factors of *2.46x*, *1.91x* and *1.81x* (all compared to OpenSSL's optimized NISTP implementation). The graph representation of three algorithms illustrate in figure 7. ECC Key generation performs better than ECDH and ECDSA at all key lengths, and is especially obvious when we increase the length of the key. Despite that the ECC does not have dedicated resources to the computationally intensive generation of prime numbers, but it is superior ECDSA in speed to produce the public/private key using comparable lengths. ECC key generation time produces linearly with key size, while ECDH, ECDSA grows exponentially. Finally the below figure conclude that ECDSA algorithm is faster than others.

B. Encryption/Decryption Time

Encryption and decryption time is mainly based on complexity of the algorithm, the processor speed, and so on. ECC with small key size provides much faster encryption/decryption as compared to others Figure 8. ECDSA is growing exponentially with the given key size and the Time of encryption/decryption in ECDH. ECC encryption time differs linearly depending on the input key size, also the decryption time remains in the exponential increase. The decryption time differs exponentially with key size for ECDH, ECDSA and it remains linear for ECC as the case with encryption.

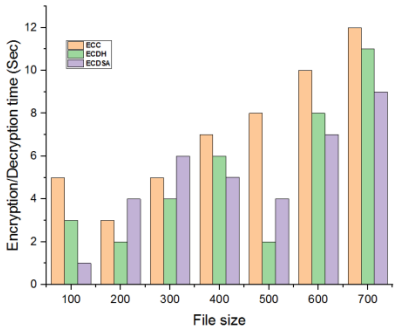

Fig. 8 Encryption and Decryption time analysis

C. Throughput

The throughput of the algorithmexecuted by split the entire information in bytes by encryption time. Higher the throughput is the efficiency of the system.

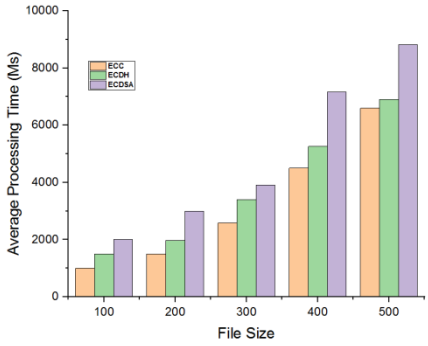

Fig. 9Throughput

The comparison between the ECDH, ECC and ECDSA is using throughput in figure 9 given below. In cryptographic algorithm, it is necessary to understand the size of output and the size of the input as this is one of the important property of an avalanche effect. Figure 9 demonstrates the throughput (in operations / second) when switching from standard OpenSSL to OpenSSL0.9.6b speed program; the corresponding precise measurements are given in figure 9. Because the OpenSSL library already contains optimized code for fixed-point multiplication, the gain is highest for random point multiplication, where throughput increases from 1600 to over 6500 operations / second.

D. Average Processing Time

The average process time each algorithm has been proposed above figure 10. The encryption and decryption time for ECDSA algorithm is less than from all other approaches (ECDSA, ECDH) in case of both for large files (10240 KB) and small files (1 KB). However ECDSA approach has the most excellent results than other approach with the intention that it is proved from the graphs even if the size of patch file size and encryption file size are large however it will take less decryption and encryption time. Therefore, ECDSA

approach is useful for achieving reduced process time of patch file on the large encrypted files.

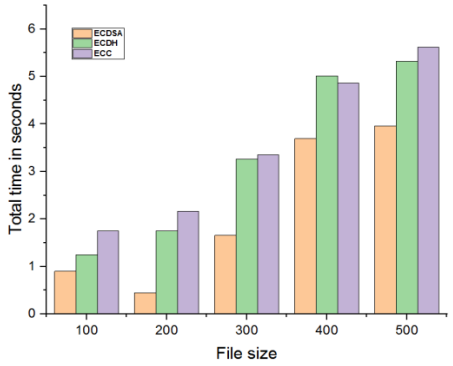

Fig. 10 Average Processing Time

VI. CONCLUSION

This survey concluded the concept of data security in cloud using encryption techniques and to check data integrity, using secure hash function algorithms. For implementation purpose, in this paper combined both encryption and Digital Signatures algorithms as a result a powerful security and data integrity service system is obtained.

It has been observed that SHA-512 and ECDSA is better than all other algorithms reviewed in this paper. SHA-512 and ECDSA is more secure than other techniques. So, the experimental results of research work show that the technique of ECDSA with SHA-512 outperforms the existing technique in terms of storage space, throughput, security and time delay.

REFERENCES

[1] Yashapalkadam,: Security in cloud Computing A Transparent View", *International Journal of Computer Science Emerging Technology*,Vol. 2,pp.316-322. October 2011.

[2] P.Metri and G.Sarote, "Privacy Issues and Challenges in Cloud Computing", *International Journal of Advanced Engineering Science and Technologies*, No. 5,pp.1-6,2013.

[3] R.Buyya, C.S.Yeo, S.Venugopal, "Cloud Computing and emerging IT platforms: vision, hype and reality for delivering computing as 5th utility", *Future Generation Computer System*, Vol. 25, pp.599-616, 2009.

[4] S. S. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic secure cloud storage with provenance," in Cryptography and Security: *From Theory to Applications, Springer*, pp. 442–464. 2012.

[5] E. M. Mohamed, H. S. Abdelkader, and S. El-Etriby, "Enhanced data security model for cloud computing," in *Informatics and Systems (INFOS), 2012 8th International Conference on IEEE*, pp.12-16. 2012.

[6] M.A. Al-Ahmad and I.F. Alshaikhli "Broad View of Cryptographic Hash Functions", *International Journal of Computer Science Issues journal (ISI Journal)*, Vol. 10, No. 4,pp.102-108, 2013.

[7] A. Al-Vahed, and Sahhavi, "An overview of modern cryptography, *World Applied Programming*, Vol. 1, No. 1,pp.55-61, 2011.

[8] A. Apostul, F. Pulcan, G. Ularu, G. Suciu and G. Todoran, "Study on advantages and disadvantages of Cloud', *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol. 2, No. 11,pp. 200-205, 2013.

[9] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, R. Konwinski, G. Lee, D. Patterson, A. Rabkin, L. Stoica, and M. Zaharia . "A View of Cloud Computing", *Comm. ACM*, Vol. 53, No. 4, pp. 50–58,2010.

[10] Ayushi, "A Symmetric Key Cryptographic Algorithm", *International Journal of Computer Applications* (0975 - 8887), Vol. 1, No. 15, pp. 1-4,2010.

[11] J. Blomer, M. Otto, and J. P. Seifert, "Sign change fault attacks on elliptic curve cryptosystems", *Fault Diagnosis and Tolerance in Cryptography*, pp. 36-52, 2006.

[12] S. K. Sood, "A combined approach to ensure data security in cloud computing," *Journal of Network and Computer Applications*, Vol. 35, No. 6, pp. 1831–1838, 2012.

[13] R. Manjusha and R. Ramachandran, "Comparative study of attribute based encryption techniques in cloud computing," *In Embedded Systems (ICES), 2014 International Conference on*. IEEE, pp.116–120,2014.

[14] B.B. Brumley and N. Tuveri, "Remote timing attacks are still practical", *Computer Security* –ESORICS, pp. 355-371, 2011.

[15] Patrick J. Flinn and James M. Jordan, "Using the RSA Algorithm for Encryption and Digital Signatures: Can You Encrypt, Decrypt, Sign and Verify without Infringing the RSA Patent", *International Journal of Computer Science and Network Security*, Vol.12 No.3, pp-82, March 2012.

[16] G. Hu, D. Xiao, T. Xiang, S. Bai, and Y. Zhang, '"A compressive sensing based privacy preserving outsourcing of image storage and identity authentication service in cloud", *Inf. Sci.*, Vol. 387, pp. 132-145, May 2017.

[17] V. Paranjape and V. Pandey, "An improved authentication technique with OTP in cloud computing", *International Journal of Scientific Research in Computer Science and Engineering*, Vol. 1, No. 3, pp. 22-26, 2013.