INVESTIGATION OF CLOUD AUTHENTICATIONS SECURITY USING POLYNOMIAL HASHING WITH SECURITY

J. Maria Selvam[1] & Dr. P. Srivaramangai[2]

[1] Ph.D. Research Scholar, Department of Computer Science, Maruthu Pandiyar College of Arts & Science, Thanjavur, Tamilnadu, India.
[2]Associate Professor, Department of Computer Science, Maruthu Pandiyar College of Arts & Science, Thanjavur, Tamilnadu, India.

Abstract: Secure data communication is of a key concern in today's rapidly growing world. Various security mechanisms are developed in order to achieve the data security. The fast growth of data volumes stored in the cloud storage has led to an increased demand for techniques for saving disk space and network bandwidth. Most of the attacker easily steal the key from the cloud server and misuse it. To prevent the unauthorized use of data accessing and create duplicate data on cloud the encryption technique to encrypt the data before stored on cloud server. Cloud backup service has become a cost-effective for storing the data as cloud is on-demand resource provider and stores data with protection. The paper presents a polynomial based novel model which uses a combination of ECC algorithm for data confidentiality, hashing for data integrity. Performance of this system is evaluated on different configurations on the basis of key, encryption and decryption time, throughput and memory usage for different data formats.

Key Words: Secure Hash Algorithm -2, Message Digest Algorithm, Polynomial Based Hashing and Elliptic Curve Cryptography, Cloud Storage, deduplication

I. INTRODUCTION

Cloud is placing important one in the research area. One main reason is that many data exhaustive application are being concerned by clouds. Cloud storage can give high availability, scalability, security, fault tolerance, cost-efficient services. For integrity, Client authentication needs security for remote login while the client's program tries to communicate with the server's program over insecure networks like Internet. The identity and a secret password of a client are used for mutual authentication and access control; password can be compromised during transmission, many authentication mechanisms were introduced to control the accessing to the cloud services depending on different authentications protocols, if an efficient scheme is not followed. So, to protect the data from unauthorized access and to ensure that our data are intact proposed a polynomial based hashing scheme, which solves the problem of integrity, unauthorized access, privacy and consistency. For confidentiality, Elliptical curve cryptography (ECC) is a public key encryption technique based on the mathematics of elliptic curves theory that can be used to create faster, smaller, and more efficient cryptographic keys. In ECC, main operations such as key agreement, signature generation, signing and verification involve scalar multiplication.

The speed of scalar multiplication plays a significant role in the efficiency of whole system. So the Fast multiplication is particularly more fundamental for some environments such as central servers, where large numbers of key agreements or signature generations occur, and in handheld devices with low computational power. Because of such importance of scalar multiplication, numerous methods have been developed, such as binary method, signed binary method, sliding window method. In this paper, use Lattice multiplication method that relies on the simple binary method. This method not solely reduces the total number of point addition, but also reduces the number of point doubling. The method is straightforward and particularly appropriate for low time delay. The objective of this paper, proposed a polynomial based novel algorithm that combines the Polynomial based Hashing for authentication and modified polynomial based ECC operations for the security mechanism. In this research aimed at designing a new security method by using a hybrid cryptosystem, for data security in the cloud. The hybrid algorithm that has been implemented is a combination of two popular and most widely used cryptographic Hashing and ECC. The new proposed polynomial based hybrid algorithm combines the PH (Polynomial based Hashing) for authentication and modified ECC operations for the security mechanism.

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.

III. 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].

Steps Overview:

- Pad message so its length plus 448 is divisible by 512.
- Appending message is end from a 64-bit value.
- Starting the Four word using 128-bit and buffer (P,Q,R,S).
- 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]:

- each round has 16 steps:
- $R(a,b,c,d,M_i,s,t_i)$: $a = b + ((a+F(b,c,d)+M_i+t_i)<<s)$
- where F(b,c,d) is a different nonlinear function in each round
- t_i is a value derived from sine

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.

- **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.
- **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.
- **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
- **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:

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

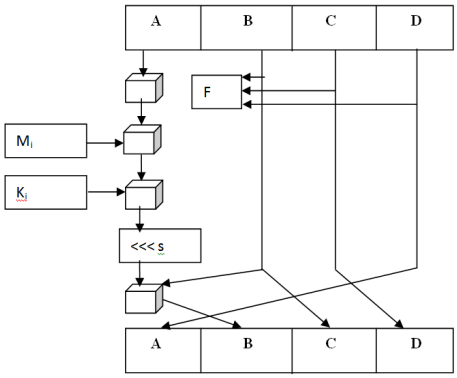


Figure 1: MD5 Processing

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.

- **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.

IV. SECURE HASH ALGORITHM -2 (SHA-2)

Secure Hash Algorithm (SHA) is described as a cryptographic hashing algorithm and this algorithm was introduced by NSA [10]. The new versions SHA-2 bear the same underlying logical binary operations, resemblance of structure, and modular arithmetic function as that of SHA-1 without sharing its weaknesses. A set of six hash functions are considered as the sequence of 224, 256, 384 or 512 bits used in SHA- 2. In favor of SHA- 256 and SHA-224 bits, each and every message block has represented 512 bits which are denoted as a sequence of 32-bit words. For SHA-512 and SHA-384, each message block includes 1024 bits as a sequence of 64-bit words. SHA-256 performs on 32-bit words and SHA-512 manages on 64-bit words. Both SHA-512 and SHA-256 are new hash functions that utilize different additive constants and shift amounts and their structures are virtually identical, varying only in number of rounds. SHA-224 and SHA-384 are the truncated versions of SHA-256 and SHA-512. SHA-512/256 and SHA-512/224 are truncated version of SHA-512. At this point, a step summary of SHA-512 is processing as given below [11]:

Steps Overview:

- SHA-512 is a variant of SHA-256 that controls on eight 64-bit words. Initially, the message to be hashed.
  - Padded with its length in such that way the result is a multiple of 1024bits long, and then the next step,
  - Parsed into 1024-bit message blocks $M^{(1)}$, $M^{(2)}$, ….., $M^{(N)}$.
  - The message blocks process one by one at a time: starting with a fixed first hash value $H^{(0)}$, evaluate sequentially as follows

$$H^{(i)} = H^{(i-1)} + C_M^{(i)}(H^{(i-1)}),$$

Where C is the SHA-512 compression function and + means word-wise mod 2^{64}addition. $H^{(N)}$ is the hash of M.

- SHA-512 applies over six logical functions. Each function is denoted as x, y, and z and controls on 64-bit words. The result of each function is formed as a new 64-bit word.

SHA example

Hash function: SHA 512

Base of Hash value: hexadecimal

File Name: example.txt

Message: Cloud computing security refers to the set of procedures, processes and standards designed to provide information security assurance in a cloud computing environment.

Hash value of original file: CF 2B 58 43 0E A0 F8 4B 92 8F 6F 82 30 12 B6 5A 12 EF F9 24 67 F3 1F F7 EB 08 FF F5 B6 EE 88 9A 3F 0A 16 1D 41 6D 73 63 B2 30 5E C0 48 C1 F5 5C 81 FA 39 16 BB 57 7A 3F 3D B4 C4 75 40 6E 05 F8

Hash value of modified file: CF 2B 58 43 0E A0 F8 4B 92 8F 6F 82 30 12 B6 5A 12 EF F9 24 67 F3 1F F7 EB 08 FF F5 B6 EE 88 9A 3F 0A 16 1D 41 6D 73 63 B2 30 5E C0 48 C1 F5 5C 81 FA 39 16 BB 57 7A 3F 3D B4 C4 75 40 6E 05 F8

Difference between the hash value of the original and of the modified file:

00000000#00000000#00000000#

phases— Initialization phase, Registration phase, Authentication phase, Encryption phase and Decryption phase. Now each of these phases is discussed below.

Initialization phase

To work with ECC, all the users must agree upon the elements defining the curve. These are called as "Domain Parameters". They are 6 distinct values: (p, a, b, G, n, h). The parameters are the base point (P), the field ($\mathbb{F}_p$), the prime number (p), the order (n), the curve (E), co-factor of curve (h), and the curve's parameters (a, b). Authority ensures that all the users agree on the domain parameters. The definition of an elliptic curve hides a lot of details: An elliptic curve is a smooth degree-3 plane curve. Basically, the elliptic curve lives in the projective plane or usual xy-plane. Every curve defined by a polynomial equation in the usual plane can be written as If f(x, y) = 0 is the equation of the curve in the xy-plane.

In this phase, several security parameters used for authentication and other processes are calculated. First, an elliptic curve equation $E_p(a, b)$: $y^2 = x^3 + ax + b \pmod p$ over a prime finite field $\mathbb{F}_p$, is selected. After agreeing on the domain parameters, The EC is generated by constructing certain polynomials based procedure and fined their roots. The N is a positive integer; N represents the degree of the polynomials at most N-1 and q is the large modulus while p is the small modulus, both of which should be relatively prime to each other. They do not have common divisor. The small polynomials f and g are secret polynomials which are selected by uniform sampling from a set of binary or ternary polynomials whereby a predetermined number of -1, 1 and 0 coefficients have been set. L_f And L_g represent private key spaces which are given by a set of small polynomials from which the private keys will be selected. The polynomials f and g are randomly generated in L_f and L_g respectively. L_m Represents the plain text space, which is given by a set of polynomials which represent the encrypt able message. The message m is expressed in the form of a polynomial whose coefficients are modulo p so that its coefficients lie between $\frac{-p}{2}$ and $\frac{p}{2}$.

Registration phase

Initially, a client A must register to the server S with his own identity ID_A and password-verifier U_A and collects the server's public key U_S , then server stores each legal client's identity, password-verifier, and a status-bit in a write protected file as depicted in the Table 3, where the status-bit indicates the status of the client, i.e., when the client is logged-in to the server the status-bit is set to one, otherwise it is set to zero.

Table 3
The verifier table.

Identity	Password-verifier	Status-bit
ID_A	$U_A = pw_A \cdot G$	0/1
ID_B	$U_B = pw_B \cdot G$	0/1
ID_C	$U_C = pw_C \cdot G$	0/1
...	...	...

Authentication Phase

The integrity is checked by comparing the hash code which is generated before and after storing data in cloud. The overall steps of the proposed model are as follows,

Authentication steps:

- User calculates the hash value for his password using a highly secure **PH (Polynomial based Hashing)** algorithm. This hash value will be used after that as a reference when it compared with the stored hash code in the server to verify user authentication attempts. Since this hash value produced from a one way function so it can't be regenerated. It is only compared with the hash value that will be generated from the server with the function.
- The user can assure that the message from authorized party. The user then sends his hash value of his password encrypted with polynomial based ECC to the server. The encryption of these parameters over the communicating channel will protect against modification or alteration from unauthorized opponent.
- The server receives the parameters from the user and starts the checking process by verifying the ID of the user from the pre stored database in the server. This assures the identity of the sender and that the message comes from authorized user.
- Server calculates the hash value and compares it with the received value from the user. Since the creation of the hash value in the server from a specific function must be matched with the created value in the server using the same function.
- Server checks the previous parameters (Checking the ID of the user and compare the received hash with the computed one using password verifier) and if the check succeeded.
- Server will generate hashing value using PH algorithm. The server will send the generated hash code encrypted with ECC to the user. The sending hash value to the user will give assurance to the user that the message is originated from the server by comparing it with the generated one.
- The user will check the received message from the server by comparing the received value with the stored one. After the check succeeded, user generates by multiplying the hash by the generating point of the curve and send to the server.

- The server also generates by multiplying the hash with the same generating point of the curve and send to the user. After exchanging the hash value between the user and the server a mutual authentication between the server and user is achieved.

Encryption and Decryption phase

The main parameters of proposed cryptosystem are integer's n, p and q. These values are used to define the following polynomial rings; the notation for the ring is given as:

$$R = Z[X]/(X^N - 1)$$

Where Z represents the set of integers and N is 1 more than the degree of the polynomial. The ring is defined such that the multiplication operation of the polynomials is wrapped around the degree ("size") of the polynomial rather than expanding the degree of the polynomial. Further, each coefficient of the polynomial is an integer that is reduced modulo certain parameters, after every math operation.

Key Generation: To create a public key, one chooses two polynomials such that $f \in L_f$ and $f \in L_g$. Here, polynomials in L_f and L_g have small widths. Also, the polynomial f should have inverses modulo p and q. In other words, one should be able to calculate f_p^{-1} and f_q^{-1} such that

$$f * f_p^{-1} = 1 \pmod p \text{ and } f * f_q^{-1} = 1 \pmod q$$

Private key is composed of the polynomials f and f_p^{-1} . After choosing the polynomials appropriately, public key can be computed as

$$h = p\, f_q^{-1} * g \pmod q$$

Encryption: Now describe how Alice wraps and sends a message to Bob using Bob's public key h. Alice chooses her plaintext m from the set. The proposed system digital envelope depends on the choice of the functions G and H, and on an integer k. The probability of forging a valid ciphertext will be P^{-k}.

$$m \in P_p\,(n-k)$$

Also chooses a random polynomialr $\in \mathcal{L}_r$. Computes

$$e = r \circledast h + [m + H(m, [r \circledast h]_p)X^{n-k} + G([r \circledast h]_p)]_p \pmod q$$

Alice then sends e to bob.

Decryption: Suppose that Bob has received the message e from Alice and
wants to decrypt it using his private key f. To do this efficiently, Bob should have precomputed the polynomial f_p^{-1} . In order to decrypt e, Bob first computes the temporary polynomial a by

$$a \equiv f \circledast e \bmod q$$

Where he chooses the coefficients of a in the interval from –q/2 to q/2. Now treating a as a polynomial with integer coefficients, bob computes the temporary polynomial $t \in \mathbb{z}[X]/(p, X^n - 1)$ by

$$t = f_p^{-1} \circledast a \pmod p$$

Further computes the two temporary quantities

$$b \equiv e - t \pmod p \text{ and } c \equiv t - G(b) \pmod p$$

And then writes c in the form

$$c = c' + c''X^{n-k} \text{ with } \deg(c') < n-k \text{ and } \deg(c'') < k$$

Finally he compares the quantities

$$c'' \text{ and } H(c', b)$$

If they are the same, he accepts c' as a valid decryption. Otherwise he rejects the message as invalid.

How to hash into elliptic curve : Some of the elliptic curve cryptosystems require hashing into an elliptic curve, for example other cryptography techniques. A particular super singular elliptic curve is used (elliptic curve scheme) in which exists a one-to-one mapping *f* from the base field $\mathbb{F}_p$ to the curve. This allows to hash using *f (h (m))* in which *h* represents a classical hash function. Moreover, the password-based authentication protocols are mainly used to provide another context in which hashing into an elliptic curve is sometimes needed. In deterministic polynomial time was issued by Shallue and Woestijne, the first algorithm mapping $\mathbb{F}_{p^n}$ into an elliptic curve [6]. The hybrid algorithms provides any elliptic curve *E* defined over $\mathbb{F}_{p^n}$, maps elements of $\mathbb{F}_{p^n}$ into *E* in deterministic polynomial time, when $p^n = 2 \bmod 3$. The algorithms depend on a rational function, explicit function from $\mathbb{F}_{p^n}$ to *E*, and that can be implemented in $O(\log^3 q)$ time and a constant number of operations over $\mathbb{F}_{p^n}$ and that technique depends on computing a cube root and simpler than the Shallue and Woestijne algorithm [7].
As an application illustrate how to hash efficiently and deterministically into an elliptic curve using two different constructions. The first construction is one-way function if the underlying hash function is one-way [8, 9]. Additionally, the second construction achieves collision resistance even if the underlying hash function is collision resistant. Given a function f into an elliptic curve *E*, explain two constructions of hash functions into *E*. Define *L* as the maximal size of $f^{-1}(P)$ where *P* is any point on *E*:

$$L = \max_{P \in E}(|f^{-1}(P)|)$$

For our encoding function $f_{a,b,}$ have L ≤ 4. Note that if work in a subgroup of *E* of order *n* with cofactor *r*, use the encoding function$f_{a,b}^{'} = r.f_{a,b}$. If *r* is relatively prime to *n*, then must have L≤ 4r. The first construction is as follows: given a hash function h:

$\{0,1\}^* \rightarrow \mathbb{F}_p$ is defined by,

$$H\,(m) = f\,(h\,(m))$$

As a hash function into the curve $E_{a,b}(\mathbb{F}_p)$. In the following, show that *H* is one-way if *h* is one way. **H**ence, the second construction given as follows. A security parameter *k* and an integer $q = p^n$ with $q \geq 2^k$, consider the following family of functions:

$$v_{c,d_{c,d\in\mathbb{F}_q}} : \{0,1\}^* \rightarrow \mathbb{F}_q \qquad x \rightarrow c.x + d$$

Where *x* is noticed as an element in $\mathbb{F}_p$. It is trouble-free to perceive that this family is 1=q^2-pair-wise independent. Given an elliptic curve *E*, merge the encoding function *f* with the functions in the $v_{c,d}$ family to get a collision-free family *G*:

$G = \left(f \circ v_{c,d}\right)_{c,d\in\mathbb{F}_q} : \{0,1\}^* \rightarrow E(\mathbb{F}_q)\ x \rightarrow f(c.x + d)$

As a final point, given a family *H* of collision-resistant functions $h: \{0,1\}^* \rightarrow \{0,1\}^k$ build the following family of hash functions into the curve *E*:

$$H_E = \left(f \circ v_{c,d}\right)_{c,d\in\mathbb{F}_q\, h\in H} : \{0,1\}^* \rightarrow E\left(\mathbb{F}_q\right)\ x \rightarrow f(c.h(m) + d)$$

ECC is a public-key cryptosystem and each user has a private key and a public key. Private Key is mainly utilized for performing with two operations such as decryption process and signature generation process. Public key is mainly used for handling two operations like encryption process and signature verification process. As an extension to other current cryptosystems are used by Elliptic curves. The security pattern of ECC is quite significant and it does not affect by the side channel attacks. Variable key lengths have been used for the encryption and are varied in reference to the data blocks to supply sufficient amount of cover the data. An elliptic curve over a field K is considered as a nonsingular cubic curve in two variables, f(x, y) =0 with a rational point (a point at infinity stage). The field K is typically taken to be the complex numbers, rational, and algebraic extensions of rational, p-adic numbers, or a finite field [12]. Key generation is an important part where we have to generate both public key and private key. The sender will be encrypting the message with receiver's public key and the receiver will decrypt its private key. To choose a number'd' handle within the range of 'n'. Using the following equation, generate the public key d = the random number and have to be chosen within the range of (1 to n-1). P is the point on the curve. The public key denote 'Q' and the private key represents 'd'.

Encryption: Let 'm' be the message is transmitted to represent this message on the curve. These have in-depth implementation details. 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

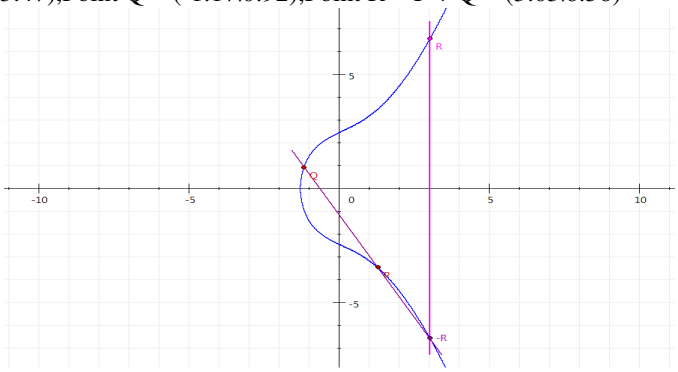
Decryption: Have to get back the message 'm' that was send

M=c2-d*c1

M is the original message that we have send.

ECC EXAMPLE

1. Curve Size: Small , Curve Type: Real number, Curve attributes: a=3, b=6, Curve: $y^2 = x^3 + 3x + 6$,Point P = (1.3|-3.47),Point Q = (-1.17|0.92),Point R = P + Q = (3.03|6.56)



2. Curve Size: Large, Curve Type: F(p), Select curve attributes: ANSI X9.62,Curve: prime192v1, Radix: 16 hexadecimal, Curve attributes: $y^2 = x^3 + 3x + 6$, where

a = ffffffffffffffffffffffffffffffeffffffffffffffc

b = 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1

p = ffffffffffffffffffffffffffffffeffffffffffffffff

Base point G: Point P

x = f0a3ab46f69d1a16f3212a9d90ece4582c8bfad13e91db82

y= 36438e499c894d67bca962ec83fb5e2fb4a2646fe4864195

Base point G: point Q

x = cdc910ab77a91d9676b6b05469ea516ae58152b1a7fec53f

y=303f7072446138caab2762f4cc991d82695ca632ba01c234

Point R : R = P + Q

x= 269f63e40da89dba0ab49f9723c77eace1e41bddb44f6a52

y= da20e0400577f70be26e8c7d0d6a8df87483ef7903b0ad25

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 = 00000006 a2adda70 3fd11169 75c07a51 801a06af 992e40f9

y= 00000007 bb784d10 6afdcf77 1b2288b2 ace3dbfd b190a5c5

Base point Q:
X=00000002 310338bb 5da9adae a49d77c9 c0a57d0b ad391353
Y= 00000005 37055f95 b9649133 f6616fde 5a9d7082 dfdc3210
Point R : R = P + Q
X= 00000006 d649e67c 8cad57b1 eb57ad15 20517342 f86aeab8
Y= 00000002 78f1fdcf db30d5f7 cc036fe6 f0600ac3 8b09f818
4. Curve Size: Small, Curve Type: F (2^m), curve attributes: m=4, f = x^4+x+1,a=1,b=1,Curve: $y^2 + xy = x^3 + x^2 + 1$, Point P = (g9|g12), Point Q = (g6|g3), Point R = P + Q = (g5|g10)

IV.EXPERIMENTAL RESULT

This section compares the performances of the scheme designed in this paper with other ECC-based authentication schemes. The implementations are done in a system having dual core Intel i7-6600U processor @ 2.60GHz. Operation system was Linux Ubuntu 16.04.The encryption and decryption algorithm are executed in the console of Google Cloud Platform using java9 and the following results are obtained. The following metrics are used to measure the performance of proposed algorithm.

Execution Time

In Figure 1, the performance evaluation of authentication time using MD5, SHA-2 and polynomial based hashing with Security (PHS) is shown. By using MD5, SHA-2 and PHS algorithms in this research work the authentication time value is 0.724µsec, 0.527 and 0.354µsec. So, the proposed algorithm betters then other algorithms.

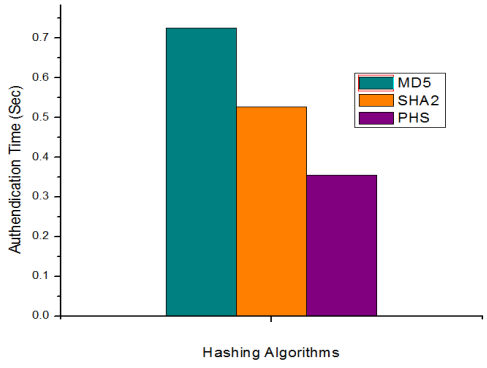


Figure 1: Comparison graph of Authentication Time with MD5, SHA-2 AND PHS

Encryption / Decryption Time (PHS)

Runtime measured in milliseconds. Graphical representation of encryption runtime is shown decryption and Decryption runtime show in Figure 2. Encryption time (ECT) is defined as the amount of time taken by the data owner to encrypt the original data into an encrypted data. Decryption time (DCT) is defined as the amount of time taken by the data owner to decrypt the encrypted data into plain data. Figure 3 shows the encryption and decryption time of the Particular Security algorithm PHS with respect to calculation.

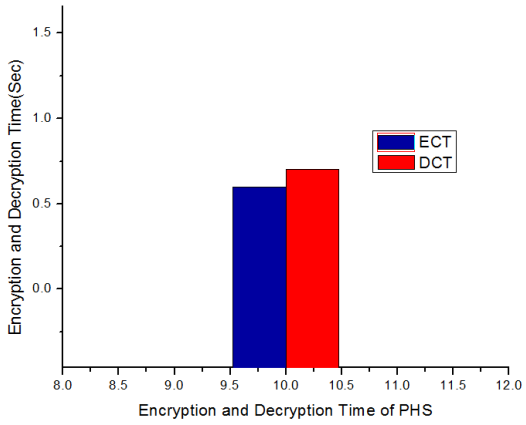


Figure 3: Encryption and Decryption Time of PHS

Throughput

Throughput is a main metric of an algorithm's performance discovering. It is a calculation that Mbits/s transferred through a network and the bandwidth capacity of the network. The throughput chart (Figure.4) shows that PHS have highest throughput. But MD5 and SHA2 are illustrated as low throughput.

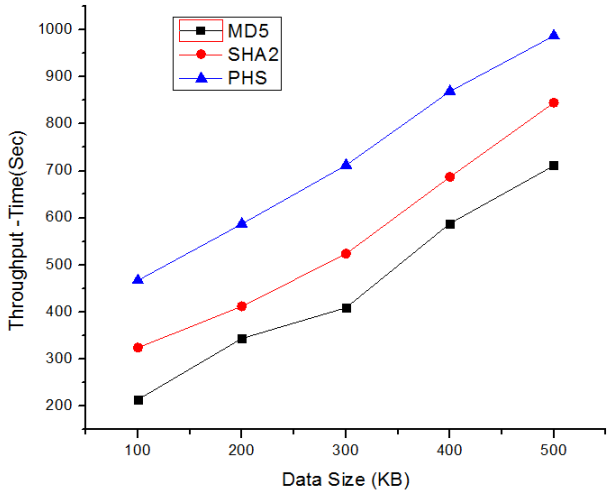


Figure 5: Throughput of MD5, SHA-2 and PHS

Memory Consumption

Memory consumption is defined as the amount of memory that is utilized for data storage in cloud and also, it is defined as an occupied capacity of CPU. In proposed algorithm we use the lightweight hash function which converts the message into bits which require less amount of memory storage. Also encryption process makes the size of message and key small.

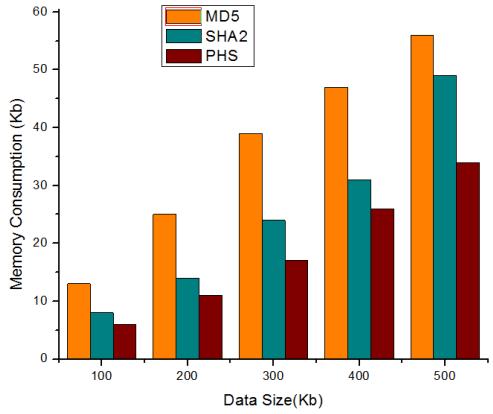


Figure 6: Comparison using Memory Consumption

Figure 6 shows the memory consumption of the proposed algorithm PHS with respect to the file size and execution time of the algorithm. From this illustration it is evaluated that the proposed algorithm requires the reduced memory space for storing the data into the cloud server.

Power Consumption

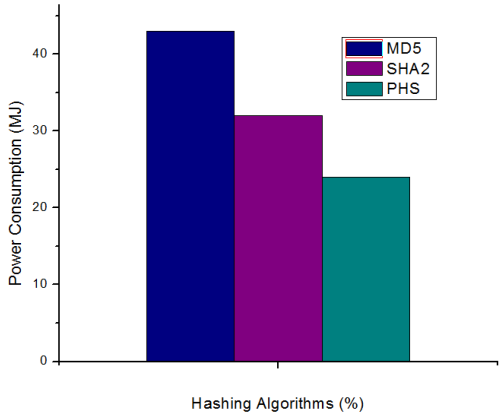


Figure 7: Power Consumption

Figure 7 shows the power consumption of proposed PHS for entire process respectively, and makes a comparison between the proposed and the protocols at different sizes of data size. The results show the superiority of PHS algorithm over other algorithms in terms of the authentication power processing using MD5, SHA, PHS algorithms; we found that PHS requires approximately 24 % of the power which is consumed for the least of other algorithms).

V.CONCLUSION

Now-a-days authentication and security has become one of the most important aspects in every field. Each information should be secured as any changes in information leads to very serious problem. Data should be secured from malicious attacks and unauthorized access. In this paper, an efficient algorithm is proposed to secure communication between sender and receiver. The algorithm first uses the lightweight polynomial based hash function for secure authentication process. Then the modified ECC algorithm is used for message and digital signature encryption and utilized for secret key encryption and decryption process. By using the hybrid cryptography algorithm the cloud computing security increases double times in terms of authentication, confidentiality and integrity. This paper concludes that PHS cryptosystem is fastest public key cryptosystems to provide different security levels at high speed. PHS provides high security solutions even on constrained devices where bandwidth, storage and computing power are limited. This work would promote confidence in both large and small scale organization in Cloud investment.

REFERENCES

[1]. P.Metri and G.Sarote (2013), ―Privacy Issues and Challenges in Cloud Computing‖, International Journal of Advanced Engineering Science and Technologies, vol.no.-5,1-6.

[2]. R.Buyya, C.S.Yeo, S.Venugopal (2009), ―Cloud Computing and emerging IT platforms: vision, hype and reality for delivering computing" as 5th utility, Future Generartion Computer System, 25: 599-616.

[3]. Schneier, Bruce.(2014) "Cryptanalysis of MD5 and SHA: Time for a New Standard". Computerworld. Retrieved .

[4]. Zhou Y, Fang Y, Zhang Y, "Securing wireless sensor networks: a survey", Journal of Communications Surveys and Tutorials, IEEE.2008 Sep; 10(3):6-28.

[5]. Ahmad Salehi S, Razzaque MA, Naraei P, Farrokhtala A,"Security in Wireless Sensor Networks: Issues and Challenges", Proceeding of the 2013 IEEE International Conference on Space Science and Communication (IconSpace), Malaysia. 2013 Jul, 356-360.

[6]. Kumar D.Sravana, Suneetha CH., Chandrasekhar A., "Encryption Of Data Using Elliptic Curve Over

Finite Fields", International Journal of Distributed and Parallel Systems (IJDPS), Volume 3, No.1, January 2012.

[7]. He D, Kumar N, Chilamkurti N, Lee JH., "Lightweight ECC based RFID authentication integrated with an ID verifier transfer protocol", Journal of Medical Systems 2014; 38(10): 116.

[8]. Leon Groot Bruinderink, Andreas Hlsing, Tanja Lange, and Yuval Yarom, "Flush, gauss, and reload a cache attack on the bliss lattice-based signature scheme", Cryptology ePrint Archive, Report 2016/300, 2016. http://eprint.iacr.org/ 2016/300.

[9]. BinuSumitra, "Secure and Usable Authentication Framework For Cloud Environment", Advances in Intelligent Systems and Computing (springer) 183-202 10.1007/978-81-322-2650-5_12 Publication date 3 Nov 2015.

[10]. SnehaK.Khodke, "Multi Level Authentication Technique for Access Organization in Cloud Data", International Journal of Advance Foundation and Research in Computer (IJAFRC) Volume 1(2348 – 4853) Publication date 12 December 2014.

[11]. Jasleen Kaur, Dr. Sushil Garg, "Security in Cloud Computing using Hybrid of Algorithms", pp. 300-305, 2015.

[12]. Manorama Chauhan, "An implemented of hybrid cryptography using elliptic curve cryptosystem (ECC) and MD5", Inventive Computation Technologies (ICICT), International Conference, 26-27 Aug. 2016

[13]. Klima, Vlastimil, Finding MD5 Collisions – a Toy for a Notebook (online), Cryptology ePrint Archive, Report 2005/075, 2006, Available at: http://eprint.iacr.org/2005/075 Accessed on: 2013-26-01.

[14]. Colin Boyd, Paul Montague, and Khanh Quoc Nguyen. Elliptic curve based password authenticated key exchange protocols. In Vijay Varadharajan and Yi Mu, editors, ACISP, volume 2119 of Lecture Notes in Computer Science, pages 487{501. Springer, 2001.

[15]. Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In EUROCRYPT, pages 156171, 2000.