Enhanced Authentication Mechanism to Protect Unauthorized Access in Public Cloud Environment

S.Hendry Leo Kanickam
Research Scholar, Department of Computer Science, Bishop Heber College, Trichy, Tamilnadu, India.

Dr. L. Jayasimman
Assistant Professor, Department of Computer Applications, Bishop Heber College, Trichy, Tamilnadu, India.

ABSTRACT

Cloud computing is a rising commercial infrastructure paradigm that assures to remove the need for maintaining and handling much expensive computing hardware. As the market develops, the threat to data also grows. To secure the data from unauthorized access and to ensure that the data are intact with proposed a scheme, and that solves the issue of privacy, integrity, consistency, and unauthorized access problems. In this research paper, proposed an Improved Polynomial based Hashing to promise the clients about their data to authenticate in the cloud. The performance of the technique is compared with other hashing algorithms.

Keywords: MD5, Secure Hash Algorithm -2, Polynomial Based Hashing, Cloud Authentication.

1. INTRODUCTION

Cloud computing assures to aid organizations, and their IT departments are more efficient and capable of cost-effective handling delivery of new services that allow their businesses to thrive [1]. However, with the growing popularity of the cloud services, the security issues have developed more noticeable, how to defend user privacy and control data from being unlawfully accessed has become a challenging issue and research hotspot. User data privacy is essential in the cloud storage for protecting the user data in the cloud server. The authentication based on classical methods relies on tokens (ID cards) and secret data that are possessed only by the user [2]. Though the traditional methods are highly accessible, they have various limitations and disadvantages [3], for example, the password of a user can be overlooked, speculated or stolen; and the user tokens can be harmed, shared, lost or stolen. If an imposter gets a user token and password or if a user shares the token and password with the imposter, then instead of the genuine user, the imposter gets authenticated and gets total access to the user's resources, as these systems are not able to differentiate between a genuine user and an imposter. These weaknesses in traditional authentication systems can be overcome by using other authentication systems. The primary step to crack these issues is user authentication, which can verify the authenticity of communication participants. A secure user authentication scheme will first authenticate the authenticity of the user when he/she request to access the cloud data; then to prevent a malicious cloud server trick users, the authority of the cloud server should be tested; once confirming the identity of the user and the cloud server, a session key will be established to encrypt the communication messages. Here, it introduces a new authentication technique by using a polynomial based hashing methodology for user confidentiality in the cloud environment. The performance of the proposed algorithm is equated with other hybrid algorithms example, MD5, SHA.

2. CRYPTOGRAPHIC HASH FUNCTION

A cryptographic is the only solution that has a hash function with the deterministic procedure that transforms any arbitrary size of the message into a fixed-size hash value or message digest. It is a one-way hash function that means it is easy to compute hash value, but it is challenging to find the original text from the hash value. The most commonly used algorithms to execute the hash function are MD-4, MD-5, SHA-0, SHA-1, and SHA-2. The cryptographic hash function is used to create digital signatures, message authentication codes (MACs), and other forms of authentication.

2.1 MD5 Algorithm

MD5 stands for Message-Digest Algorithm 5. It is one of the most widely used cryptographic hashing algorithms in the field of authentication security[4]. It was introduced by professor Ronald Rivest [5] in the year 1992. It is an improved version of MD4. It takes the various length of input and generates a fixed length of 128 bit as output. MD5 has less CPU intensive and less complicated than other hash functions. For instance, where MD5 is a commonly used approach in the general non-top-secret applications. The weakness of MD5 is reported vulnerable to a hash collision [6]. This weakness permits the malicious users can create multiple input sources to MD5 that result in the hash collision occurrence.

2.1.1. MD5 Algorithm Procedure

MD5 takes an arbitrary length of input and generates a fixed length of 128 bit as output. Steps involved in MD5 procedure are listed as follows

Step 1: Add padding bits and length field to the original message

Step 2: The message from step 1 is partitioned into blocks of 512 bits: B1, B2, ..., Bn.

Step 3: The message blocks are processed by a compression function C as follows:

Vk = C(Vk-1,Bk); k= 1,2,. . . . n

Where Vk-1 and Bk are the compression function input values, and Vk is the output value, which is 128 bits long. In order to start the process, the initial value V0 is provided by the MD5 standard.

Step 4: Finely, the message digest is given by Vk.

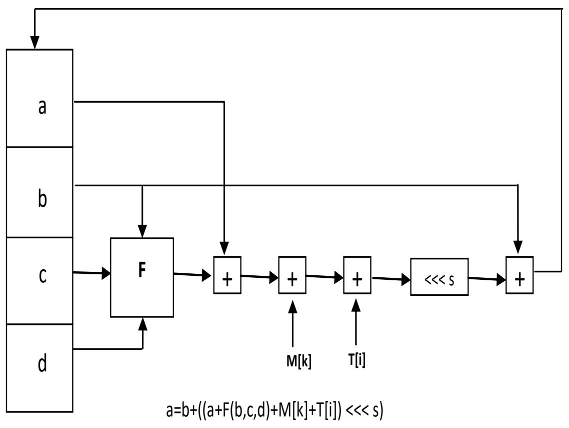
MD5 algorithm works as follows:

1. Padding bits and a 64-bit length field are added to the original message in order to make the resultant message as multiple of 512 bits (represented as M)
2. The resultant message (M) divided into series of 512 bits blocks (B1, B2, ..., Bn).
3. Divide 512 bit block (B) into 16 blocks(X1, X2 . . .X16) of 32 bits in size.
4. Initialize Channing variables
   A, B, C, D chaining variables which are represented in four 32 bit Shift registers and these are initialized by following hexadecimal number
   A = 0x67452301
   B = 0xEFCDAB89
   C = 0x98BADCFE
   D = 0x10325376
5. The A, B, C, and D values are copied into four different variables as AA, BB, CC, and DD, respectively.
6. Define 4 Logical functions (F,G,H,I)
   F, G, H, I are four basic MD5 functions. Each function takes three 32-bit words as input and generates one 32-bit word as output. The algorithm involves four rounds, and each round consisting of 16 steps. The total no. of steps is 64.

Round	**Function**
1	$F(b, c, d) =(b \wedge c)\vee ((\neg b) \wedge d)$
2	$G(b, c, d) = (b \wedge d)\vee ((c \wedge (\neg d))$
3	$H(b, c, d) = b \oplus c \oplus d$
4	$I(b, c, d) = c \oplus(b \vee (\neg d))$

7. These four rounds utilize the same operation structure but use a different function, namely F, G, H, I for each round, respectively. The four operations are defined as follows.

FF(a, b, c, d, M[k], s, i): a=b+((a+ F(b,c,d)+M[k]+T[i]<<<s);D=C;C=B;B=A;A=D
GG(a, b, c, d, M[k], s, i): a=b+((a+ G(b,c,d)+M[k]+T[i]<<<s);D=C;C=B;B=A;A=D
HH(a, b, c, d, M[k], s, i): a=b+((a+ H(b,c,d)+M[k]+T[i]<<<s);D=C;C=B;B=A;A=D
II(a, b, c, d, M[k], s, i): a=b+((a+ I(b,c,d)+M[k]+T[i]<<<s);D=C;C=B;B=A;A=D



M[k], 0≤k≤15 represents the kth sub-block of the message block(B) and <<<s represents left shift operation

8. At last 4th rounds, the output is added to the input of first round
   A=A+AA; B=B+BB; C=C+CC; D=D+DD
9. All of these steps are completed, then the algorithm is continued to run the next 512-bit message block. Finally, the message digest produced 128 bit as output, which is the concatenation of values in A, B, C, and D.

2.2 Secure Hash Algorithm -2 (SHA-2)

The Secure Hash Algorithm (SHA) is described as a cryptographic hashing algorithm, and this algorithm was introduced by NSA [7]. The new versions SHA-2 bear the same underlying logical binary operations, the 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 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 the 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 versions of SHA-512. At this point, a step summary of SHA2 is processing as given below [8]:

Steps Overview:

- SHA-512 is a variant of SHA-256 that controls 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 in sequentially as follows
  H(i) = H(i-1) + CM(i)(H(i-1)),
  Where C is the SHA-512 compression function and + means word-wise mod 264addition. 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: SHA2
The 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#00000000#00000000#00000000#00000000#00000000#00000000
#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#0000000
0#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#000000
00#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00000
000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#0000
0000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#000
00000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00000000#00
000000#00000000#
```

0.00% of the bits differ (0 of 512).
Most extended identical bit sequence: offset 0, length 512.

3. IPH (Improved Polynomial Hashing)

The research aims to explain a new hash algorithm using polynomials over finite fields. The hash has many attractive characteristics in terms of its flexibility. Specifically, the length of the hash is a parameter that can be set at the outset. Additionally, the estimated degree of collision resistance is computed in terms of another parameter to compare whose value may be differed. The hash function that constructs has the following essential attributes. Initially, the length of the output can be modified simply by changing a few steps of the calculation. Secondly, the computation performance is evaluated as a bit-stream procedure to oppose a block procedure. Finally, many features of the construction, which are the Current Register (CR) construction, the compression function, and the exponentiation and truncation, appear to be novel constructs. The construction uses polynomials over finite fields. Note that earlier efforts have used polynomials over finite fields in the construction of hash algorithms. However, the make use of polynomials is very different.

Authentication steps:

- The user calculates the hash value for the password using a highly secure IPH (Improved Polynomial Hashing) algorithm. This hash value will be used after that as a reference when compared with the stored hash code in the server to verify user authentication attempts. Because this hash value produced from a one-way function so that it cannot be regenerated. It is compared only with the hash value, which will be produced from the server with the function.
- The user can assure that the message from the authorized party. The user then sends his hash value to the server. The encryption of these parameters over the communicating channel will securely protect against alteration or modification from the unauthorized opponent.

- The server receives the parameters from the user and begins the checking process by verifying the ID of the user from the pre-stored database in the server. It assures the identity of the transmitter and that the message arrives from the authorized user.
- The server computes the hash value, and it compares with the received value from the authorized user. Because the creation of the hash value in the server from a specific function should be matched with the generated value in the server using the same function.
- The server checks the previous parameters (Checking the ID of the user and matches up to the received hash with the computed one) and if the check succeeded.
- The server will create hashing value using the IPH algorithm. The server will transmit the created hash code to the user. The hash value is to be transmitted to the user, and that will assure 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, the user generates by multiplying the hash by the generating point of the curve and send it 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, mutual authentication between the server and the user is achieved.

To initiate three constructions may be of interest in their own right. The algorithm, then, can be described in brief as below.

- Initially, the CR construction is obtained as input a sequence of k polynomials over F_2 of degree < n and produces another such sequence.
- Secondly, the compression routine receives as input a binary sequence of length k, a sequence of k polynomials over F_2 of degree < n, an integer r with 2n < r ≤ k, and a sufficiently large integer λ, and generates r matrices of 2n rows and 1 + λ columns. This construction is invertible. In other words, given the matrices, one can reconstruct both the binary sequence in addition to the sequence of polynomials. The compression function is obtained by eliminating columns 2 to 1+λ of selected rows of these matrices.
- The third construction is truncation, followed by exponentiation in a finite group. The hash function obtains a message specified as a binary string of length k and executes a preliminary operation on it to convert it into a sequence of k polynomials over F_2 of degree < n. After that process, it invokes the CR construction and the compression routine, respectively.

In the Cantor enumeration, the entries of the compressed matrices are combined to generate a single integer. In the truncation-exponentiation routine, this integer is used to generate a hash value.

Enumeration:

Finally, a single integer generates from the vectors Ri using Cantor enumeration. For each d ≥ 1, there is an adjective map

$$e_d : \mathbb{N}^d \rightarrow \mathbb{N}$$

For d = 1, we can take the identity, and for d = 2, we can take

$$e(x,y) = e_2(x,y) = \frac{(x+y)^2 + 3x + y}{2}$$

Given e*n*, the map

(x_1,…..,x_(n+1)))→e_n (e_2 (x_1,x_2),x_(3),…,x_(n+1))

It is a candidate for en+1. On the other hand, there are more optimal choices. To apply the enumeration, function to the vectors Ri and need the functions em where m = irr(n) for four ≤ n ≤ 10. first record the values of the pairs (n,m) in Table 1. Explicit candidates that produce numbers of manageable size for n = 4, 5, 6 are given in the appendix. For example, for n = 4, we used

Table 1: Number of irreducible polynomials

N	4	5	6	7	8	9	10
M	7	10	16	25	43	71	129

e_7 (s_1,s_2,s_3,s_4,s_5,s_6,s_7)=e(e(e(e(s_1,s_2) s_3),e(s_4,s_5)),e(s_6,s_7))
This number is the order (k/10)16. Especially, for a 1MB file, this is about 40 bytes.

Truncation-exponentiation:

The third of the "primitives" (the first two being the CR construction and the compression function) is known as truncation followed by exponentiation in a group and is explained as follows. Let

$$H: \{0,1\}^* \to \{0,1\}^k$$

A hash function with output length κ. Let G is considered to be a finite abelian group and select an element g ∈ G.

Let

$$F: G \to \{0,1\}^T$$

be a function with τ < κ. For a string M ∈ {0, 1}*, consider the new function

$$H: M \to F\left(g^{int(H(M))}\right) \in \{0,1\}^T$$

Here, int is the function that associates to a bit string the integer that it represents in base 2.

Algorithm:

PARAMETERS: e, $n_1, \dots, n_e, \{r_j, s_j, g_j, q_j\}, T$

INPUT: Message *M* of length *k*

OUTPUT: Hash value *H* of *M* of T bits

1. Compute the stretching and splitting $s(M, n_j)(1 \le j \le e)$
2. Calculate the masking $CR_i^{(n_j)}$ for $1 \le j \le e$ and $1 \le i \le k$.
3. Compute the tables $T_i^{(n_j)}$ for $1 \le j \le e$ and $1 \le i \le r_j$. Each table has 2^{n_j} entries, and each entry has s_j bits.
4. Compute bit strings and their associated integers $BS_i^{(n_j)}$
5. From the tables, compute the spectra $R_i^{(n_j)}$ and their associated integers $C(R_i^{(n_j)})$.
6. To utilize both sets of integers to evaluate an integer I_j (for $1 \le j \le e$).
7. Compute H_j in the group.
8. The final hash value *H* is the sum of the $H_j$in-group *G*.

4. EXPERIMENTAL RESULT

Experimental verification of comparison of MD5, SHA2, and IPH algorithms is carried out using below Machine configurations: i5 processor and 4 GB RAM. The simulation has been undertaken on java and obtained the results of the performances of hash algorithms. The experiment has been taken place for various text files. Performances of algorithms calculated in the basis of,

- **Time Analysis**

The message digest calculation time is one of the essential parameters while observing the performance of any algorithm. The observed time is in seconds. Figure 1 shows the result of the time taken by each algorithm to generate a hash value for different text files. It is seen that IPH generates hash value, everything else in a short time. Furthermore, SHA2 and MD5 are taking a long time to generate hash values than proposed algorithms.

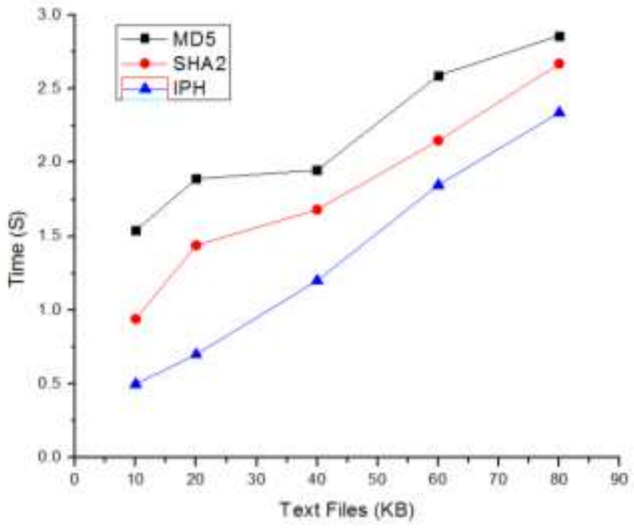


Figure 1: Time consumed by various hash algorithms

- **Throughput**

Throughput is the primary metric of an algorithm's performance discovering. It is a calculation that MB/S transferred through a network and the bandwidth capacity of the network. The throughput chart (Figure.2) shows that IPH has the highest throughput. However, MD5 and SHA2 are illustrated as low throughput.

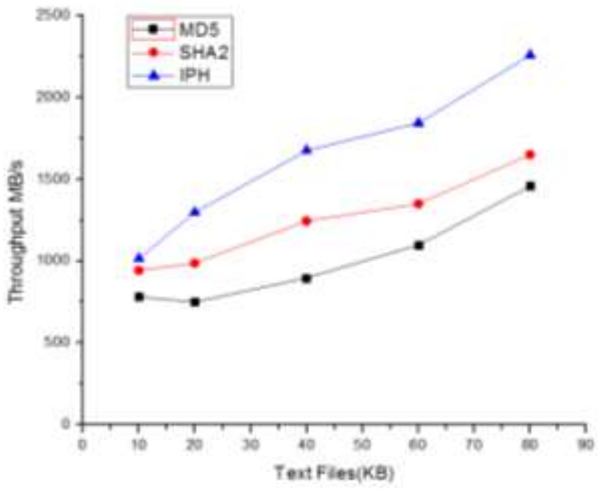


Figure 2: Throughput

5. CONCLUSION

Cryptographic hash functions are essential in the current communication scenario. Many algorithms have been used to generate message digest or message authentication codes. As the computational speed and attacks are increasing day by day, MD5 and SHA2 are no longer called as a secure and efficient message-digest algorithm. In this paper, proposed a new secure hash algorithm Improved Polynomial Hashing (IPH). By analyzing MD5, SHA2, and IPH, we conclude that IPH is faster as compared to others. Moreover, by utilizing the results of this analysis, one can make a system that uses these hashing algorithms together to make a more secure and efficient system.

REFERENCES

[1] Pritesh Jain, Prof. Vaishali Chourey and Prof. Dheeraj Rane,” An Analysis of Cloud Model-Based Security for Computing Secure Cloud Bursting and Aggregation in Real Environment” published in International Journal of Advanced Computer Research (IJACR), ISSN (Print): 2249-7277, ISSN (Online): 2277-7970 Volume 1, pp. 23-28, September 2011.

[2] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas. Robust and Reverse-Engineering Resilient PUF Authentication and KeyExchange by Substring Matching. IEEE Trans. on Emerging Topics in Computing, 2(1):37–49, 2014.

[3] K. Ntalianis and N. Tsapatsoulis. Remote Authentication via Biometrics: A Robust Video-Object Steganographic Mechanism Over Wireless Networks. IEEE Trans. on Emerging Topics in Computing, 4(1):156–174, 2016

[4] Wang, Xiaoyun, Feng, Dengguo, Lai, Xuejia, and Yu, Hongbo, Collision for Hash Functions MD4, MD5, HAVAL-128, and RIPEMD (online), Cryptology ePrint Archive, Report 2004/199, 2004, Available at http://eprint.iacr.org/2004/199 Accessed on 2013-26-01.

[5] Rivest, Ronald, The MD5 Message Digest Algorithm (online), Internet Engineering Task Force, 1992, Available at: http://tools.ietf.org/html/rfc1321 Accessed on: 2013-25-01.

[6] 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.

[7] 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.

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