International Innovative Research Journal of Engineering and Technology

ISSN NO: 2456-1983

International Conference on Emerging Innovation in Engineering and Technology

ICEIET-2017

Enhanced Public Integrity Auditing On Cloud Data Using Sha Algorithm

R. Sumathy[1], P. Kanchanadevi[2] , Dr.A.Amuthan[3]

[1]Pondicherry Engineering College, Puducherry, India
[2]Pondicherry Engineering College, Puducherry, India
[2]Pondicherry Engineering College, Puducherry, India
[1]sumathy0177@gmail.com
[2]kanchanaperumal@pec.edu
[3]amuthan@pec.edu

Abstract

Cloud storage growth promotes the secure remote data auditing as a blistering topic. It deals with the problem of security and efficiency of public data integrity auditing during dynamic data sharing. The existing scheme provides the proficient public integrity auditing with secure group user communication based on group signature, Asymmetric Group Key Agreement (ASGKA) and vector commitment. But still this scheme is not consistent for secure group user revocation and also for dynamic cipher text database. In collusion attack cloud is able to learn the contents of shared data colluding with untrusted or revoked user. This paper proposes Secure Hash Algorithm-2 (SHA-2) with dynamic user management that supports dynamic cipher text and efficient user revocation. Secure Hash Algorithm-2 concerns about the intended group communication in that any member can leave or join the group at any time. It also has a set of cryptographic hash functions that prevents collusion. Additionally this work wrapped up with the properties, such as confidentiality, efficiency, countability and traceability of public data integrity auditing**.**

Keywords- Asymmetric Group Key Agreement (ASGKA), Secure Hash Algorithm (SHA), Public Data Integrity Auditing, Vector Commitment**.**

I. INTRODUCTION

Cloud computing is the growth of a different technologies that have come together to change an organization's method to construct an IT infrastructure. The term cloud computing explains a different types of computing ideas that involve a huge number of computers connected via real time communication network [1]. The enlargement of cloud computing motivates many enterprises and organizations to share their data with the Cloud Service Providers (CSPs), that will limit the usage of local devices. Recently, some commercial cloud storage services, such as the Simple Storage Service (S3) [19] on-line data backup services of Amazon and some practical cloud based software [20-24], have been built for cloud application. In some cases the cloud servers may provide an inaccurate result, such as server hardware/software failure, human maintenance and malicious attack [25]. So a new form of methods are required to achieve the data integrity and accessibility that supports to protect the security and privacy of cloud user's data.

In favor of providing the integrity and availability of remote cloud store, some solutions [26, 27] and their variants [28, 29] have been proposed. In these solutions, when a system supports data alteration, we name it dynamic scheme, or else static one. A System is publicly verifiable, that the data integrity check can be executed not only by data owners, but also by any unauthorized auditor. However, the dynamic designs above focus on the cases where there is a data owner and only the data owner could change the data.

In recent time, the development of cloud computing lifted some applications [30-32], where the cloud service is used as a cooperation platform. In these developing environments, multi users in a group need to share the data in order to make changes such as modify, delete, update, run and compile the source code at any time. The new collaboration network model in cloud makes the remote data auditing methods become impossible, where only the data owner can update their data. Insignificantly extending a scheme with an online data owner to update the data for a group is irrelevant for the data owner. It will cause enormous communication and computation overhead to data owner, which will result in the single point of data owner. To carry multiple user data process, B. Wang et al. [33] proposed data integrity based on ring signature. In this method, the user revocation drawback is not considered and the auditing cost is linear to the group size and data size. To enhance the previous scheme and support group user revocation, B. Wang et al. [34] designed a scheme based on proxy re-signatures. However, this scheme concluded that the private and authenticated channels exist between each pair of entities and there is no collusion resistance among them. Also, the auditing cost of the scheme is linear to the group size. s The data owners designed polynomial authentication tags and adopt proxy tag update systems in their scheme, which make their scheme support public checking and efficient user revocation. However, in their methods, the creator does not care about the data secrecy of group users. It means, their scheme could economically support plaintext data update and integrity but not ciphertext data. In their scheme, if the data owner insignificantly shares a group key among the group users, the defection of any group user will force the group users to update their public key. In addition, the data owner does not take part in the user revocation module, where the admin authority itself could conduct the user revocation phase. In this instance, the collusion of revoked user and the cloud server will give probability to third-party cloud server, where the cloud server could able to update the data as designed and supply a legal data finally. Still there is no proper solution for public integrity auditing with group user modification. By using pervious secure symmetric encryption scheme data owner in the group needed to use only the random secret key and then encrypt the data. To support multi-user data modification, such that to share the data at the same time. The shared encrypted data and a shared secret key among group users will lead to single point failure issues. The revoked user will break the data confidentiality. To avoid the above problem, Tao Jiang designed Asymmetric Group Key Agreement scheme (ASGKA) that supports multi-user data modifications [4]. It means that to share encrypted data among dynamic groups and also to have verifiable database update. This scheme uses only a shared encryption key instead of a common secret key. Thus, the public key is used for both signatures verification and message encryption, as long as any signature that comes under this public key can be used to decrypt ciphertext. But ASGKA has drawbacks such that it doesn't support user's authentication, dynamic management of group users and there is also collusions between the cloud server and the database is possible. To overcome the above problems we proposed the SHA- 2 to implement the public auditing system which is used to produce digital signature for the data to verify the authenticity of the user's signature. The Messages Digest is created that represented in the hexadecimal format by using the hash functions. Thus in the group user only with the valid digital signature can read, share, update, and or delete the data. Thus the security goals are archived by the collusion property, assumptions of security analysis and for database updates, vector commitment is used.

1.1 OUR CONTRIBUTION

In this paper, we further study the problem of construing public integrity auditing system for dynamic group management on the shared dynamic data. Our contributions as follows:

1. We explore efficient public auditing on multi-users with dynamic group management on the shared data.
2. By incorporating both Vector Commitment and SHA- 2, Collusion Resistance is achieved.
3. Our proposal securely verifies the users authentication and data integrity is managed by this scheme.
4. Additionally security goals are achieved such that countability and traceability.

1.2 RELATED WORK

In the existing work Asymmetric Group Key Agreement Protocol (ASGKA) has public encryption key and a unique decryption key is generated for each group members with respect to its public encryption key. This protocol is used for multi-user modifications in the group member communications [4]. Jiawei Yuan et al proposed a novel integrity auditing concept for cloud data sharing services which is characterized by the following such that multi-user modification, public auditing, high error detection probability, efficient user revocation and along with the practical computational/communication auditing performances. This scheme avoids user impersonation attack. This scheme is associated with Batch auditing of multiple tasks [13]. Dipali S. Kasunde et al [14] proposes the system where shared data integrity of multi-owners are verified. This scheme helps to determine the collision attack that happens when the revoked user try to communicate with the cloud storage server. Thus the proxy multi-signature scheme is developed to verify data integrity. Ram Krishna Dahal et al [15] illustrates SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed to hide the data from unauthorized user. While sharing the data in the cloud environment, SHA is used to encrypt/decrypt the data by using the hash functions of this algorithm. Xiaofeng Chen et al [10] describes the verifiable database with incremental updates (Inc-VDB). This framework is implemented for data updates of a large amount of data. It's incorporated with basic concepts of vector commitment and the encrypted version of incremental MAC mode is again encrypted. Inc-VDB framework based on the computational Diffie-Hellman (CDH) assumption. Mingchu Li et al [5] proposed the identity-based public-key cryptography (IB-PKC) to authenticate users with their public key signatures. It's framework resistant active attacks and overcomes all the security issues. Additionally this scheme allows any users to join or leave the group at the same time. Cong Wang et al [16] proposed to utilize and uniquely combine the public key based homo-morphic authenticator with random masking to implement the privacy-preserving public cloud data auditing system that supports to effectively handle multiple auditing tasks. Boyang Wang et al [17] proposed a novel public auditing mechanism for the shared data integrity. By implementing proxy re-signatures, allows the cloud to re-sign blocks for all the existing users during user revocation, so these users need not to download and re-sign blocks by themselves. However, a public verifier can always able to determine the shared data integrity without retrieving the entire data from the cloud. Even though, a part of shared data has been already re-signed by the cloud storage server. Jiawei Yuan et al [13] propose a novel data integrity checking scheme that supports for data sharing among multi- writers. By using this framework a constant size of data integrity proof need to be transmitted to the public verifier, irrespective of data blocks or writers are associated with the data blocks. This framework has obvious advantages in terms of efficiency, scalability and security. Yong Yu et al [18] illustrate a practical public integrity auditing framework associated with multi-user data modification.

1.3 Organization

Our paper organization as follows: In section 2, we describe the problem formulation. In section 3, we discuss the core preliminaries. Then, we include the detailed explanation of our proposed scheme in section 4. We determine the security analysis and performance evaluation in section 5. Finally, we discuss our conclusion in section 6.

II. PROBLEM FORMULATION

In this section, we first explain the cloud storage model of our system. Then, we provide the threat model and also the system model contemplated.

2.1 Cloud Storage

The cloud storage model consists of three entities, namely the cloud storage server, group users and a Third Party Auditor (TPA). The cloud storage server is a semi-trusted which promotes data storage for the group users. The group users consist of data owner and a set of users who are authorized to access and modify the data or file by the data owner. TPA is an entity in the cloud, who will examine the data integrity of the shared data stored in the cloud storage. In our system, the data owner is responsible to encrypt and upload into the cloud server. Also, the data owner shares the privilege such as access and modify with the group users. Even frequently updated data by the group users is also verified by the TPA to maintain the data integrity. The data owner securely revokes the group user when the group user is found malicious or when the contract of the user is terminated.

2.2 Threat Model

In our threat model there are two types of attack exist: 1) An attacker outside the group or revoked user may seize some of the plaintext of the data. Actually, these kinds of attacker at least try to break the security of the group data encryption scheme. 2) The cloud server may collude with the revoked group users, and the revoked user tries to provide an illegal data without being detected.

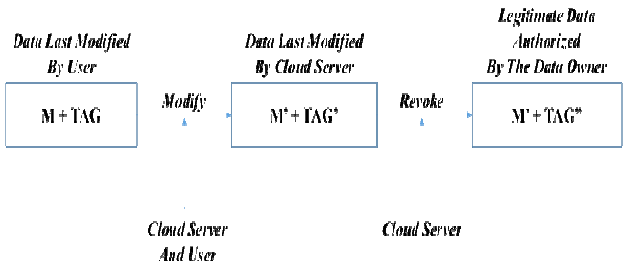


Figure 1: Security Problem of Server Proxy Group User Revocation

Figure 1 shows Security Problem of Server Proxy Group User Revocation. Actually, we presume that the cloud storage server is semi-trusted. Thus, it's prudent that a revoked user will collude with the cloud storage server and share their secret group key with the cloud server. Although the server proxy group user revocation [2] reduces the cost of communication and computation, but this scheme makes malicious cloud storage server to get the secret key of revoked users. Thus, this malicious cloud server changes the last modified user data m into malicious data m′. During user revocation process, the cloud could make the malicious data m′ become valid. To overcome the above problems, we aim to achieve the following security goals:

1. ***Security-*** A scheme is secure, if any data in the database are not able to access by the attacker and any adversary are not able to give an invalid data to the verifier.
2. ***Correctness-*** A scheme is correct, if any valid user update the data m and encrypted value is stored in cloud storage server.
3. ***Efficiency-*** A scheme is efficient, if the computation and storage space used by any client user must be independent of shared data size.
4. ***Countability-*** A scheme is countable, if Third Party Auditor (TPA) is able to provide the misbehaviour proof of the data and the dishonest cloud server that has been tampered with the database.
5. ***Traceability-*** A scheme is traceable, if the data owner is able to track the last user updates (who update the data items), and every signature generated by the users.
6. ***Authenticity-*** A scheme is acceptable, if the data owner is able to verify the group user with their group signature is valid.

2.3 Secure Hash Algorithm (Sha)

Secure Hash Algorithm (SHA) is a cryptographic hash function that creates a 160-bit (20-byte) hash value which is called "Message Digest". The hash values are in hexadecimal format and 40 digits long. There are different versions of SHA as follows [3]:

SHA-0: The original version of 160-bit hash function.

SHA-1: A 160-bit hash function which is similar to the earlier MD5 algorithm.

SHA-2: A family has two similar hash functions and with different block sizes, called SHA-256 and SHA-512. Both are different in the word size; SHA-256 uses 32 byte words where SHA-512 uses 64 byte words. The truncated versions of each standard called as SHA-224, SHA-384, SHA-512/224 and SHA-512/256.

A, B, C, D and E - 32-bit words of each states

F - Nonlinear Function which varies

$<<<_r$ - Left Bit Rotation at position r

r - Changes for each operation

W_t - Expanded Message word at round t

K_t - Round Constant of round t;

⊞- Addition of Modulo 2^{32}.

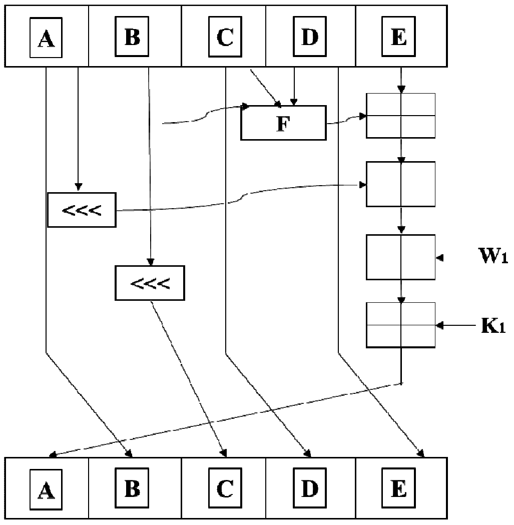


Figure 2: shows the SHA-1 Compression Function- One iteration

2.4 System Model

This system model encompasses of Secure Hash Algorithm- 2 (SHA- 2), Dynamic Management and Vector Commitment concepts. Thus we can resolve the public integrity data auditing problems associated with security issues.

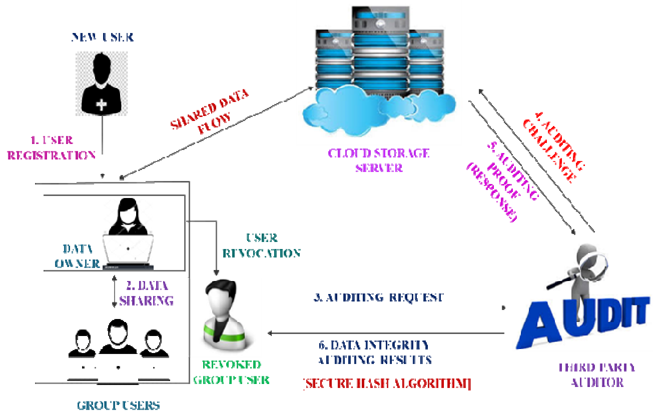


Figure 3: System Model of Public Auditing System

Figure 3 describes System Model of the proposed Public Auditing System that consists of the three main entities [4]

- Group of multiple users
- Public verifier
- Cloud storage

Group consists of multiple users that share the data with each other. It consists of manager or owner of a group that able to generate security parameters which is used to eliminate revoked users. Remaining users of the group are already registered users they will store the data into the cloud server as well as shares the data among themselves. Public Verifier or Third Party Auditor (TPA) who utilizes the shared data for specific purposes and promotes verification services on the integrity of the data. SHA is implemented for key generation and encryption/decryption of all the data while sharing among the group members. And a perfect session time is established for each member in the group, so their communication will be collision resistant. Thus the dynamic group communication is maintained. Then the hash functions will create the encryption/decryption of all the data that in turn able to manage the data integrity in the database.

II. Preliminaries

3.1 Bilinear Group

A group G is a set of elements that consists of two elements X, Y ∈ G to form another element Z ∈ G under the operation $\circledast$, denoted by $X \circledast Y=Z$. The group operation to form an additive group, say addition "+", such as forms an elliptic curve. The group operation to form a cyclic multiplicative group, say multiplication "×", such as forms a primitive residue classes modulo and denoted as XY. Now we move to the concept of bilinear maps. Let G_1 be an additive group of prime order q and G_2 be a cyclic multiplicative group of the same order q. And let Q be a generator of G_1.

A bilinear map $ê: G_1 \times G_1 \rightarrow G_2$ follows these properties:

- ***Bilinearity:*** Given A, B, C $\in G_1$, we have $ê(A+B,C) = ê(A,C) \cdot ê(B,C)$ and $ê(A, B+C) = ê(A, B) \cdot ê(A,C)$
- ***Non- degeneracy:*** $ê(Q, Q) \neq 1$
- ***Computability:*** There is an efficient algorithm to execute ê(A, B) for any A, B $\in G_1$.

3.2 Complexity Assumption

A function μ(x): $N \rightarrow \mathbb{R}$ is called negligible for every positive integer c, there exists an integer N_c such that $\forall x > N_c$, $|\mu(x)| < 1/x^c$. Thus the negligible function will become rapidly approach zero as the argument increases. So the probability of negligible function can be avoided.

There are some computational problems and complexity assumptions employed in our protocol. Let q, G_1, G_2, Q, and ê: $G_1 \times G_1 \rightarrow G_2$ be defined as previously mentioned. Our security assumptions as follows:

A. *Definition- Computational Diffie-Hellman (CDH) Problem:* Given xQ, yQ for some unknown x, y $\in Z^*_q$, to compute xyQ.

B. *Definition- Bilinear Diffie-Hellman (BDH) Problem:* Given xQ, yQ, zQ for some unknown x, y, z $\in Z^*_q$, to compute $ê(Q, Q)^{xyz}$.

C. *Definition- k-Bilinear Diffie-Hellman Exponent (k-BDHE) Problem:* Given H $\in G_1$ and $\{R_i = x^iQ\}_{i \in \{1,\ldots,2k\}, i \neq k+1} \in G_1$, where $x \in Z^*_q$ is unknown, to compute $ê(Q, H)^{k+1}$.

Assumptions- CDH (BDH, k-BDHE)
Still there is no proper solution to solve the polynomial-time problem of CDH (BDH, k- BDHE) with non-negligible probability.

3.3 Collusion Resistant Hash Function (Crhf)

The formal definition of CRHF was given by I. Damgard [6] [7] and an informal definition were given by R. Merkle in [8].

Definition- A collision resistant hash function is a function h satisfying the following conditions:

1. Function h must be publicly known and does not acquire any secret information for its operation (extension of Kerckhoffs's principle).
2. The argument X has an arbitrary length and the result of h(X) gives a fixed length of n bits where $n \geq 128$.
3. Given h and X, the computation of h(X) must be "EASY".
4. The hash function should be one-way. It means given Y in the image of h, its "HARD" to determine a message X such that h(X) = Y. Given X and h(X) and its "HARD" to determine a message $X' \neq X$ such that $h(X') = h(X)$.
5. Thus the hash function must be collision resistant means its "HARD" to find two different messages that holds the same hash value.

3.4 Vector Commitment

In cryptography, commitment is a fundamental strategy and it plays a crucial role in security protocols such as voting, identification, zero-knowledge proof, coin flipping etc. The commitment has two type of properties. Such that hiding and binding properties. The hiding property of commitment concerns that it should not publish information of the committed message, and the binding property concerns that the committing structure should not be changed by the sender [4].

Lately, Catalano and Fiore [9] promoted a new primitive called Vector Commitment. Vector Commitment satisfies position binding that an enemy should not be able to open a commitment to two different values at the same position. It means that the commitment string size and its openings have to be independent with respect to the vector length. The formal definition of Vector Commitment as follows [10]

Definition- A vector commitment VC = (VC.KeyGen, VC.Com, VC.Open, VC.Veri, VC.Update, VC. Proof Update) consists of the following algorithms

VC.KeyGen(1^l, q): Input- security parameter l and size q = poly(l) of the committed vector. The key generation algorithm gives public parameters PP that implicitly defines the message space M.
VC.Com$_{pp}$($m_1, \cdots, m_q$): Input- a sequence of q messages $(m_1, \cdots, m_q) \in M^q$, and public parameters PP. A commitment string C and an auxiliary information aux is generated by committing algorithm.
VC.Open$_{pp}$(m, j, aux): This algorithm is executed by the committer to generate a proof π_j that m is the j-th committed message.
VC.Ver$_{pp}$(C, m, j, π_i): The verification algorithm generates 1, if and only if π_j is a valid proof that C is a commitment to a sequence $(m_1, \cdots, m_q)$ such that $m = m_j$.
VC.Update$_{pp}$(C, m, j, m'): This algorithm is executed by the original committer who changes the j-th message to m' while updating C. Input- Old message m at the position j,

new message m'. Output- A new commitment C' along with an update information U.

VC.ProofUpdate$_{pp}$(C, U, m', j, π_j): Any user who holding a proof π_j executes this algorithm for some message at the position i with respect to C. It also allows the user to figure out an updated proof π'_j of updated commitment C' such that π'_j is valid with respect to C' which contains a new message m' at the position j.

3.5 Sha- 512 Steps

A. Pre-Processing

In SHA-2 input block size rely on the algorithm used. Here our input block size is 1024-bit. The pre-processing stage first divides the original message into N chunks or blocks, namely M(1), M(2) ,…, M(N). Each chunks has 1024 bits. Then message padding must be carried out, if the message length is not a multiple of the considered block size. Then a set of eight initial hash values, $H_0^{(0)}$,…, $H_7^{(0)}$ are listed. Figure 4 describes the each algorithms implementation that gives a different set of initial hash values [11].

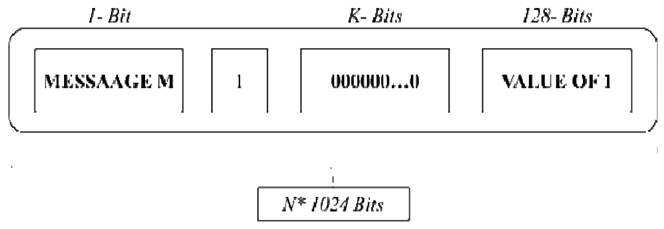


Figure 4: *Message pre-processing (SHA- 512)*

B. Hash Computation

The hash computation operation is based on input block size 1024-bit words. The number of iterations executed by the algorithm is given by m = 80. Actually, m is assumed to represent the number of 1024-bit words processed by the algorithm [12]. More specially, the SHA-512 algorithms composed of m message schedule words (W_0 ,…, W_{80}), eight active working variables (a, b, c, d, e, f, g, h), and eight intermediate hash values ($H_0^{(l)}$,…,$H_7^{(l)}$). SHA-512 uses 80 constants (K_0 ,…, K_{80}), where each one of them is 1024-bit size. Furthermore, six binary logical functions are used together, as shown below. ROR n(x) and SHR n(x) represents a rotation and a shift of x by n bits to the right. Additionally ^, $\oplus$, and ~x represents the bitwise operation of AND, XOR and complement of x.

In SHA-512 the hash computation step implements four logical functions: Ch, Maj, Σ_0, and Σ_1. The result of each new function is a new 64-bit [11]:

$$Ch(x, y, z) = (x \wedge y) \oplus (\sim x \wedge y)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (z \wedge y)$$

$$\sigma_0(x = OTR_1(x) \oplus ROTR_8(x) \oplus SHR_7(x)$$

$$\sigma_1(x) = ROTR_{19}(x) \oplus ROTR_{61}(x) \oplus SHR_6(x)$$

$$\Sigma_0 = ROTR_2(x) \oplus ROTR_{34} \oplus ROTR_{39}\ (x)$$

$$\Sigma_1 = ROTR_{14}(x) \oplus ROTR_{18} \oplus ROTR_{41}(x)$$

For each message chunk l, $1 < l < N$, a four-step digest round is executed and as follows [11]

Initialize the eight working variables:
$a = H_0^{(l-1)}$, $b = H_1^{(l-1)}$, $c = H_2^{(l-1)}$, $d = H_3^{(l-1)}$
$e = H_4^{(l-1)}$, $f = H_5^{(l-1)}$, $g = H_6^{(l-1)}$, $h = H_7^{(l-1)}$

Prepare the message schedule:
$W_l = M_0^{(l-1)}$; $0 \leq m \leq 15$
$W_l = ROTL_1\ (W_{m-3} \oplus W_{m-8} \oplus W_{m-14} \oplus W_{m-16})$; $16 \leq m \leq 79$

SHA-512 algorithm is given:

```
For t = 0 to 79:
{
$Q_1 = h + \Sigma_1^{256}\ (e) + Ch\ (e, f, g) + K_m^{256} + W_l$
$Q_2 = \Sigma_0^{256}\ (a) + Maj\ (a, b, c)$
$h = g$
$g = f$
$f = e$
$e = d + Q_1$
$d = c$
$c = b$
$b = a$
$a = Q_1 + Q_2$
}
```

Compute the l^{th} intermediate hash value $H^{(l)}$:
$H_0^{(l)} = a + H_0^{(l-1)}$, $H_1^{(l)} = b + H_1^{(l-1)}$,
$H_2^{(l)} = c + H_2^{(l-1)}$, $H_3^{(l)} = d + H_3^{(l-1)}$,
$H_4^{(l)} = e + H_4^{(l-1)}$, $H_5^{(l)} = f + H_5^{(l-1)}$,
$H_6^{(l)} = g + H_6^{(l-1)}$, $H_7^{(l)} = h + H_7^{(l-1)}$,

After, processing all N blocks of message M, the final message digest is achieved by concatenating the hash values ( ). The concatenation of words is represented by the symbol ‖.

$$H_0^{(N)} \parallel H_1^{(N)} \parallel H_2^{(N)} \parallel H_3^{(N)} \parallel H_4^{(N)} \parallel H_5^{(N)} \parallel H_6^{(N)}$$

IV. Proposed Scheme

SHA-2 consists of various algorithms such that SHA-224, -256, -384 and -512. Here we consider SHA -512 algorithm for the implementation of secure public integrity auditing system. SHA-512 algorithm composed of 3 stages:

- Message Padding
- Block Expansion
- Round Computation

A. Stage 1- Keys and Hash Code Generations

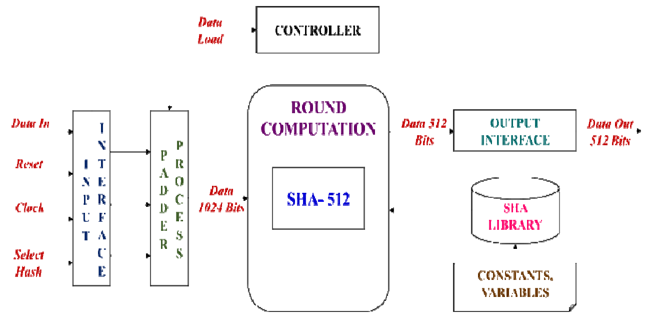


Figure 5: Processing of SHA-512

Figure 5 shows the proposed configured SHA- 512 algorithm. The given architecture accomplishes four operation modes for reconfigurable SHA processor [11].

Input Interface and Output Interface: During the data loading process, both input interface and output interface have to buffer the input and output data or messages.

***Control Unit*:** The control unit is designed to control the flow of data and as well as data exchange between the Padded Procedure Unit and Hash Computation Unit.

Hash Computation Unit: The Hash Computation Unit is a data path component of the system architecture. SHA-512 performs 80 cycles to obtain 512-bit. For each cycle, SHA-512 algorithm need the previous rounds, and as well as the constant value Ki. This core uses eight 64-bit words: a-d, words are initialized to the predefined values, during each hash function calls.

Padded Process Unit: Padded Process Unit adds the input messages and converts it to 1024-bit.

Figure 6 shows the Finite State Machine (FSM) of padding process.

- FSM executes the following five states:
- Pad 0: Packets of 8- bits of data are recovered at each stage.
- Pad 1 and Pad 4: Transmission of 512- bit packets are counted followed by compatibility checking.
- Pad 2: Finds the number of ‘0’bits to be added.
- Pad 3: Last packet is loaded by the binary coded message.

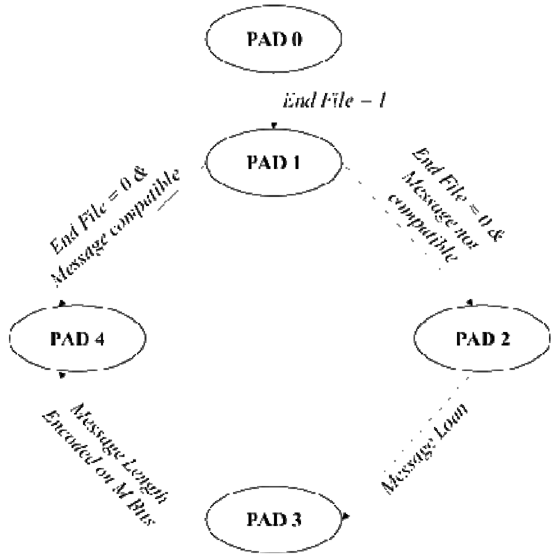

PAD 0
End File – 1
PAD 1
End File = 0 & Message compatible
End File = 0 & Message not compatible
PAD 4
PAD 2
Message Length Encoded on M Bits
Message Load
PAD 3

Figure 6: Finite state machine of the padded process

B. Stage 2- Session Establishment

This stage consists of set of functions as follows:

Set-SV(): This function is executed by the users. Gives user identity U as input and the session value of U is generated. Session values changes at each stage.

Agree(): This function is executed by group user who want to establish a session. To execute it, each user must provide the input- the session information, params, session value, and private key. And broadcast the outputs to the others in the group.

Derive-GPK(): This function is executed by any sender. Given input- session information, params, and the broadcast messages of the group users. Then group encryption key is obtained.

Derive-GDK(): This function is executed by any user of the group. Given input- session information, params, the broadcast messages of the group users, and the private key of the user. Then group decryption key is obtained.

*Enc():*Executed by the sender. Given input-params, group encryption key, and the plaintext to be encrypted. Finally ciphertext of the plaintext is generated.

*Dec():*Executed by the group users. Given input- params, group decryption key, and ciphertext to be decrypted. Finally plaintext corresponding to the cipher text is generated.

C. Stage 3- Dynamic Management

There are three set of functions involved as follows:

Join(): Any newcomers can join a group by executing this function. Given input- params and the session information. Then he/she broadcasts the output message to the other users in the session. Simultaneously, the existing group users executes this same function with input as params and the session information. Then they also broadcast their output message to all the session users. Once the broadcasted messages are valid, then the newcomers are accepted by the group members.

Leave(): Any group member can leave their group. The remaining members of the group executes this function. Given input- params and session information. Finally message is broadcasted to all users in the session.

JoinWithLeave(): This function is executed by the remaining members and some newcomers. Any member can join or leave the group.

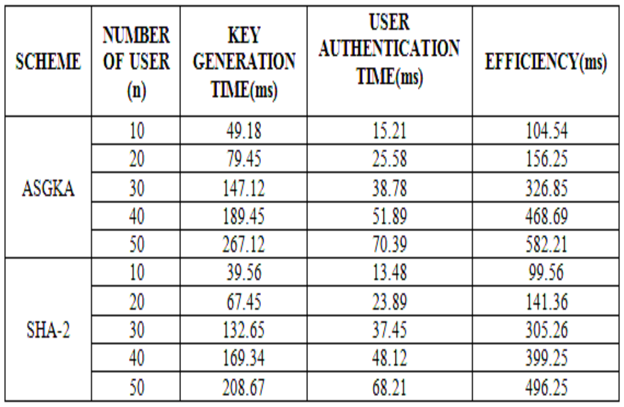
propose the efficient vector commitment algorithm for the frequent data updates. By using the hiding and binding property of the vector commitment [10].

III. Security Analysis And Performance Evaluation

A.Numerical Analysis: In this segment, we numerically analyze our scheme and compare with in terms of key generation, user authentication, and efficiency times are displayed in Table 1. Then data update times are given in Table 2.

TABLE 1

Comparison of ASGKA and SHA- 2



SCHEME	NUMBER OF USER (n)	KEY GENERATION TIME(ms)	USER AUTHENTICATION TIME(ms)	EFFICIENCY(ms)
ASGKA	10	49.18	15.21	104.54
	20	79.45	25.58	156.25
	30	147.12	38.78	326.85
	40	189.45	51.89	468.69
	50	267.12	70.39	582.21
SHA-2	10	39.56	13.48	99.56
	20	67.45	23.89	141.36
	30	132.65	37.45	305.26
	40	169.34	48.12	399.25
	50	208.67	68.21	496.25

B. System Setup

We first assess the generation of public keys, master keys and secret keys for the system. Our result in Figure 7(a) indicates that the key generation cost is proportional to the number of user, since the master user needs to generate secret keys for each group user separately. To show the performance of authentication time generation, we vary the number of users in the application is from 1 to 50. As shown in Figure 7(b), the authentication generation time increases proportionally to the number of users, from 15.21ms to 70.38ms. And Figure 7(c), shows the Efficiency time is proportional to the number of user.

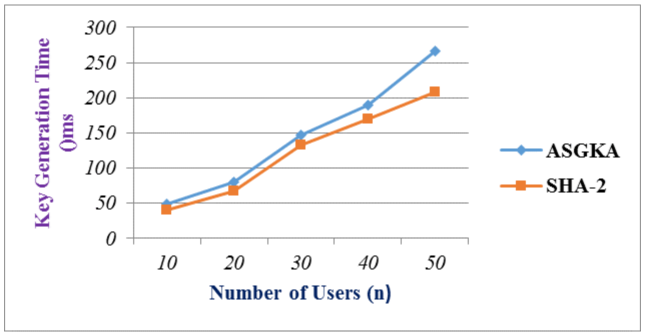


Figure 7(a): Key Generation time (ms)

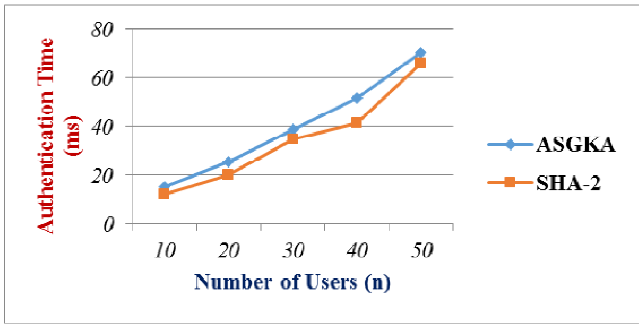


Figure 7(b): Authentication time (ms)

Figure 7(c): Efficiency (ms)

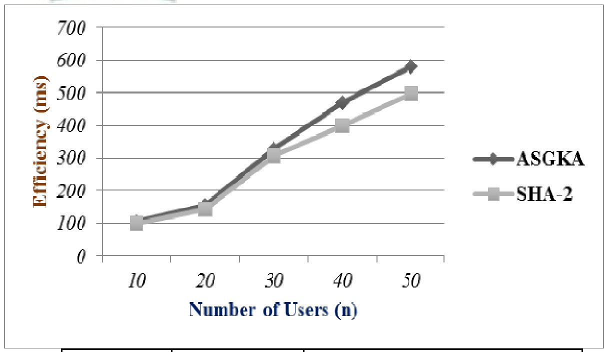
C. Data Update

The update procedure of data items in our scheme is similar to the setup procedure. As shown in Figure 8, the data update time is proportional to the number of modified data items.

TABLE 2

DATA UPDATE

Scheme	Number of ta items (n)	Data updation time(ms)
ASGKA	10	98.89



	20	139.45
	30	180.81
	40	224.41
	50	363.78
SHA-2	10	80.31
	20	127.12
	30	156.25
	40	210.98
	50	322.90

VI. CONCLUSION

Cloud storage provides the most prominent services for many organizations. Though, the cloud storage has different advantages, it has problems regarding the integrity and the security of shared data in the cloud. The primitive of verifiable database with efficient updates is an

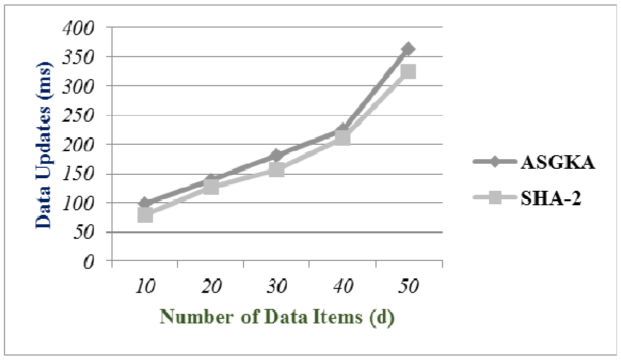


Figure 8: Updation time (ms)

important procedure to solve the problem of stored data. In the existing system, they promoted a scheme to have efficient public auditing system. This scheme composed of Vector Commitment, Asymmetric Group Key Agreement (AGKA) and group signatures with user revocation are adopted to achieve the data integrity auditing of remote data. ASGKA that supports multi-user data modifications. It means that to share encrypted data among dynamic groups and also to have verifyable database update. This scheme uses only a shared encryption key instead of a common secret key. Thus, the public key is used for both signatures verification and message encryption, as long as any signature that comes under this public key can be used to decrypt ciphertext.

Beside the public data auditing, the combining of the three primitive enable our scheme to outsource cipher text database to remote cloud and support secure group users revocation to shared dynamic data. When multiple users are working in a group, there should be a mechanism to revoke the users. In collusion attack cloud is able to learn the contents of shared data colluding with untrusted or revoked user. We determine the method to avoid collusion attack with efficient revocation of users while verifying the integrity of data shared by users. Thus ASGKA has drawbacks such that it doesn't support user's authentication, dynamic management of group users and there is also collusions between the cloud server and the database is possible.

To avoid the above problems, we propose a proficient public integrity auditing with secured group user communication based on Secure Hash Algorithm (SHA-2). SHA-2 designed for compatibility with increased security provided by the Advanced Encryption Standard (AES) cipher. It is used to produce digital signature for the data to verify the authenticity of the user's signature. The Messages Digest is created that represented in the hexadecimal format by using the hash functions. Thus in the group, user only with the valid digital signature can

read, share, update, and or delete the data . It has a set of cryptographic hash functions that prevents collusion. It is used for both signature generation and verification. It also provides constant security for the group member's communication. Additionally dynamic user management is managed such that any member can leave or join the group. Thus the security properties, such as confidentiality, efficiency, countability and traceability of secure group user revocation are achieved. Finally, in the comparison of experimental analysis reduces the security complexity using this proposal.

This will be useful in many applications such as in PHR (Patient Health Record) system where more than one doctors treating a patient can share sensitive information with each other. This system can also be useful in Version Control Systems (VCS) which is widely used for automation of documentation, configuration files and management of source code for developing software. Proposed system can be used to verify the integrity for VCS.

VII. REFERENCES

[1] Joseph K. Liu, Kaitai Liang, Willy Susilo, Jianghua Liu, Yang Xiang, ”Two-Factor Data Security Protection Mechanism for Cloud Storage System” IEEE Transactions on Computers, 2015.

[2] J. Yuan and S. Yu, “Efficient public integrity checking for cloud data sharing with multi-user modification,” in Proc. of IEEE INFOCOM 2014, Toronto, Canada, Apr. 2014, pp. 2121–2129.

[3] SHA. (2005) SHA. Google. [Online]. Avialable: https://en.wikipedia.org/wiki/SHA-1#SHA-0/

[4] Tao Jiang, Xiaofeng Chen, and Jianfeng Ma,”Public Integrity Auditing for Shared Dynamic Cloud Data with Group User Revocation,” IEEE Transaction on Computers, 2015.

[5] Mingchu Li, Xiaodong Xu, Cheng Guo and Xing Tan, “AD-ASGKA – Authenticated Dynamic Protocols for Asymmetric Group Key Agreement,” in Security and Communication Networks, January 2016, Wiley Online Library, pp.1340-1352.

[6] J. Daemen, R. Govaerts, and J. Vandewalle, “A hardware design model for cryptographic algorithms,” Computer Security – ESORICS 92, Proc. Second European Symposium on Research in Computer Security, LNCS 648, Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, Eds., Springer-Verlag, 1992, pp. 419–434.

[7] I.B. Damgard, “Collision free hash functions and public key signature schemes,”Advances in Cryptology, Proc. Eurocrypt 87, LNCS 304, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 203–216.

[8] R. Merkle, “One way hash functions and DES,” Advances in Cryptology, Proc. Crypto 89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428–446.

[9] D. Catalano and D. Fiore, “Vector commitments and their applications,” PKC 2013, LNCS 7778, Springer-Verlag, pp.55-72, 2013.

[10] Xiaofeng Chen, Jin Li, Jian Weng, Jianfeng Ma, Wenjing Lou, “Verifiable Computation over Large Database with Incremental Updates,” IEEE Transactions on Computers, 2015.

[11] Hassen Mestiri, Fatma Kahri, Belgacem Bouallegue, Mohsen Machhout, “Efficient FPGA Hardware Implementation of Secure Hash Function SHA-2” Published Online on I.J. Computer Network and Information Security, 2015.

[12] SHA- 2. (2010) SHA- 2. Google. [Online]. Avialable:https://www.vocal.com/cryptography/sha-algorithm/

[13] Jiawei Yuan, Shucheng Yu, “Public Integrity Auditing for Dynamic Data Sharing with Multi-User Modification,” IEEE Transaction on Information Forensics and Security, 2015.

[14] Dipali S. Kasunde, A. A. Manjrekar, “Verification of Multi-Owner Shared Data with Collusion Resistant User Revocation in Cloud,” IEEE Transaction on International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), 2016.

[15] Ram Krishna Dahal, Jagdish Bhatta, Tanka Nath Dhamala, “Performance Analysis of SHA-2 and SHA-3 Finalists,” International Journal on Cryptography and Information Security (IJCIS), September 2013, Vol.3, No. 3.

[16] Cong Wang, Qian Wang, Kui Ren and Wenjing Lou, “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,” in the Proceedings of IEEE INFOCOM, 2010.

[17] Boyang Wang , Baochun Li and Hui Li, “Public Auditing for Shared Data with Efficient User Revocation in the Cloud,” in the Proceedings of IEEE INFOCOM, 2012.

[18] Yong Yu, Yannan Li, Jianbing Ni, Guomin Yang, Yi Mu and Willy Susilo, “Comments on Public Integrity Auditing for Dynamic Data Sharing with Multi-user Modification”, IEEE Transactions on Information Forensics and Security, 2015.

[19] Amazon. (2007) Amazon simple storage service (amazon s3). Amazon. [Online]. Available: http://aws.amazon.com/s3/

[20] Google. (2005) Google drive. Google. [Online]. Available: http://drive.google.com/

[21] Dropbox. (2007) A file-storage and sharing service. Dropbox. [Online]. Available: http://www.dropbox.com/

[22] Mozy. (2007) An online, data, and computer backup software. EMC. [Online]. Available: http://www.dropbox.com/

[23] Bitcasa. (2011) Inifinite storage. Bitcasa. [Online]. Available: http://www.bitcasa.com/

[24] Memopal. (2007) Online backup. Memopal. [Online]. Available: http://www.memopal.com/

[25] M. A. et al., "Above the clouds: A berkeley view of cloud computing," Tech. Rep. UCBEECS, vol. 28, pp. 1–23, Feb. 2009.

[26] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. of ACM CCS, Virginia, USA, Oct. 2007, pp. 598–609.

[27] A. Juels and B. S. Kaliski, "Pors: Proofs of retrievability for large files," in Proc. of ACM CCS, Virginia, USA, Oct. 2007, pp. 584–597.

[28] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: theory and implementation," in Proc. of CCSW 2009, llinois, USA, Nov. 2009, pp. 43–54.

[29] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," in Proc. of ACM CCS 2013, Berlin, Germany, Nov. 2013, pp. 325–336.

[30] Cloud9. (2011) Your development environment, in the cloud.Cloud9. [Online]. Available: https://c9.io/

[31] Codeanywhere. (2011) Online code editor. Codeanywhere. [Online]. Available: https://codeanywhere.net/

[32] eXo Cloud IDE. (2002) Online code editor. Cloud IDE. [Online]. Available: https://codenvy.com/

[33] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in Proc. of IEEE CLOUD 2012, Hawaii, USA, Jun. 2012, pp. 295–302.

[34] B. Wang, L. Baochun, and L. Hui, "Public auditing for shared data with efficient user revocation in the cloud," in Proc. Of IEEE INFOCOM 2013, Turin, Italy, Apr. 2013, pp. 2904–2912.

Author Biography

R. Sumathy received her B.E. degree (2015) in Computer science and Engineering. Currently, she pursues M.tech (2017) in Pondicherry Engineering College. Her research interest includes Cloud computing and Web service.

P. Kanchanadevi received her B.Tech (2014) in Computer science and Engineering. Currently, she pursues M.tech (2017) in Pondicherry Engineering College. Her research interest includes Cloud computing and data mining and search Engine optimization.