International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 12, No.1 (Dec-2022)

https://dx.doi.org/10.12785/ijcds/1201120

An Efficient and Secure Auditing System of Cloud Storage Based on BLS Signature

Ghassan Sabeeh Mahmood[1], Noor Hasan Hassoon[2], Hazim Noman Abed[1], Baidaa Abdulrahman Jalil[1]

[1] *University of Diyala, College of Science, Computer Science Department, Diyala, Iraq*
[2] *University of Diyala, College of education for Pure Science, Computer Science Department, Diyala, Iraq*

Received 28 Mar. 2021, Revised 26 Jul. 2022, Accepted 24 Aug. 2022, Published 31 December 2022

Abstract: Cloud computing users can use the cloud storage service remotely by enjoying various applications and services upon request from a group of cloud computing resources, all without the burden of storing data as well as maintaining it. However, cloud users can no longer possess data and this makes data integrity an important issue. Also, users should use cloud storage as if it is locally, without thinking about verifying the integrity of cloud data. To solve the problems and to provide auditing of cloud storage securely and effectively without additional burden on users, this paper proposes an efficient and secure auditing system of cloud storage that supports auditing and maintains users' privacy based on Boneh-Lynn-Shacham (BLS) signature. Besides, the proposed system worked with dynamic data integrity verification mechanisms as well. The safety and performance analysis of the system has proven the effectiveness and safety of the proposed system.

Keywords: Data Integrity, Cloud Computing, BLS Signature, Data Auditing

1. INTRODUCTION

The Cloud paradigm is considered as a promising model to enable access to the network for a set of computing resources like networks, servers, various services, different applications, as well as remote storage, which can be provided quickly and with minimal effort of the cloud service provider. That is why cloud computing services are considered very important after the main services such as electrical utilities, gas and water services, and telecom services. In recent times, business organizations and users have relied on cloud computing services in terms of document creation and data backup, in addition to cloud-based email services. That is why cloud computing is an important and effective way for organizations and users, as a result of the transformation directly and indirectly to the cloud paradigm, the information technology expenditures will be affected by one trillion dollars in the next few years [1].

Despite all the benefits offered by the cloud computing model, it suffers from security problems. As an example, a database failure in Salesforce caused data to be lost for three and a half hours permanently [2]. On the other hand, a Dropbox security breach caused 68 million user accounts to be leaked. Once data is outsourced to the cloud, this results in a lack of control over the data and thus puts data integrity at risk. There are many reasons for risking the integrity of data. Therefore, the cloud computing service must deal with malfunctions and problems in software as well as hardware that may lead to the destruction of user data. The provider of cloud service may choose to hide errors in the data to disguise them. In order to provide good storage space, a cloud service provider (CSP) may try to store data that is rarely accessed in offline ways, or this data may be deleted. Therefore, these reasons invite users of cloud computing to use effective and robust methods for performing data integrity audits frequently [3].

With the time development of information technology in the recent era, many systems have been introduced [4,5] that verify the integrity of the data stored in the cloud by accessing the entire file in order to ensure the integrity of the data. However, these systems will be less efficient with large files, because the verification process will take a long time. In addition, it may lead to the user being restricted to the number of tries to check file integrity. Other schemes [6,7,8,9,10] have been developed to design remote data integrity verification protocols, this enables checking cloud data integrity without downloading it completely from the cloud. The system

E-mail: ghassan.programer@gmail.com, noor.hassan@puresci.edu.iq,hazim_numan@sciences.uodiyala.edu.iq, scicompms04@uodiyala.edu.iq.

proposed by [6] is a public checking scheme with confidentiality in the environment of cloud computing however is not safe. The main purpose this system is not secured is that it is highly susceptible to attacks from a bad server or external attacker. The system proposed by [9] guarantees remote data integrity verification but it does not support dynamic processes of the data. Ge et al. [8] proposed a system that would ensure data integrity verification and data retrieval through various servers, but their system's scope is restricted only to fixed data. Whereas Erway et al. [7] suggested a system for auditing the data integrity that supports dynamic data processes. They used a skip list-based data structure to support modifications to cloud data, and the efficiency of this system remains unidentified. Therefore, due to users' lack of trust in cloud computing and the problems mentioned in the above systems in terms of insecurity, inefficiency and accessing the whole file to guarantee integrity, in addition to supporting static data only or not supporting batch auditing, these factors are what motivated us to propose this paper.

In this regard, this paper proposes a secure and efficient cloud data storage auditing system based on a BLS signature to periodically check the integrity of cloud data without completely restoring data or by reducing additional online burdens for cloud users. In addition to ensuring that privacy and confidentiality of data are maintained using the AES encryption algorithm. In addition, the suggested scheme has expanded the scope of work by supporting dynamic data operations, so that users are constantly updating their data in the cloud environment. To maintain the security of storage safety at the same level, the proposed system has worked effectively and safely, and this has been proven by security and performance analysis. Thus our contribution can be highlighted as follows:

- The authors suggest a remote data integrity verification scheme depending on the BLS signature that ensures secure and efficient data auditing, privacy-preserving, and data dynamic operations.
- The proposed system provided support for data confidentiality in the cloud and this was done by using the AES algorithm.
- After the security analysis of the proposed system, it was evaluated from the security point of view and proved its security in a random oracle model assuming the stability of the Computational Diffie–Hellman (CDH) problems. The performance analysis also showed the effectiveness of the proposed system. The connection cost of the proposed system was also calculated and was O(n).

The rest of the paper is structured as follows. In Section 2, the related work is presented. In Section 3, a problem statement is presented. The proposed system is presented in Section 4. The evaluation of the proposed system is discussed in Section 5. Lastly, the conclusion of the paper is offered in Section 6.

2. RELATED WORK

Several auditing systems for data integrity have been recently suggested to allow only the possessor of data to perform a data integrity audit [7,8,9]. So that these systems include the data owner and the cloud storage server, and these systems are called private auditing systems. While many systems allow third-party auditors (TPA) to carry out data integrity audit, these systems are called public auditing systems [11,12,13,15].

Ateniese et al. [14] suggested an original system for data integrity proof in which their system relied on RSA signatures in addition to random sampling methods. They proposed two models to have Proof of Data Possession (PDP). The first is the sampling PDP and the second is the efficient PDP. The S-PDP model provides secure data acquisition, but at the same time, it is less efficient than the second model. The limitation in their system is only suitable for static data files.

Ge et al. [8] supported private auditing by suggesting exploring search investigation using keywords over encrypted data with symmetric key-based validation. They designed a new cumulative authentication signature depending on symmetric-key to create a cumulative authentication signature for every keyword. The disadvantage of this system does not support data dynamically. Also, supported the proposed scheme Lu et al. [9] Proving the integrity of static data only by a special audit of the data sharing system in the cloud computing environment for mobile devices. The system also ensures authorized access to data by security checks before sharing data with users to avoid incorrect calculations. Their system also achieved lightweight mobile operations on both the data owner and data requester sides. It lacked support for dynamic data operations.

Whereas F. Wang et al. [10] suggested a dynamic demonstrable data possession system with non-repudiation in the cloud computing environment by using index tables. This scheme resisted may attack and avoids the synchronization issue. Moreover, in dynamic operations, this system had lower storage cost and computation cost. However, this system cannot protect privacy. Erway et al. [7] suggested an auditing system for data integrity that supports dynamic data processes. In their cloud system, they used the skip list data structure for data modifications, but the efficiency of their cloud system is unclear.

Shane et al. [21] suggested an identity-based system for cloud data integrity auditing. In their system, files saved in cloud computing can be shared and utilized by others provided that the sensitive information about the file is secured. Although their system provides an auditing

protocol, it does not support data dynamics nor does it provide a solution to data confidentiality.

We believe that most previous work has focused on the issue of data integrity auditing, while little attention has been paid to ensuring that data integrity is protected while supporting dynamic data operations. Our work contributes to designing a data integrity verification system in the cloud environment that is secure in the data audit process and maintains privacy while supporting dynamic processes. On the other hand, it is effective in reducing user computation costs during the system preparation stage as well as in the data integrity verification stage.

Table 1 illustrates the comparison of the integrity between the proposed system and other systems in terms of data auditing, privacy preservation, confidentiality and data dynamic.

TABLE I. COMPARISON OF THE INTEGRITY SYSTEMS

Schemes	Data auditing	Privacy preservation	Confidentiality	Data dynamic
[7]	Yes	No	Yes	Yes
[8]	Yes	Yes	No	Yes
[9]	Yes	Yes	No	No
[10]	Yes	No	No	Yes
[11]	Yes	Yes	No	Yes
[12]	Yes	Yes	No	Yes
[13]	Yes	Yes	No	Yes
[14]	Yes	Yes	No	No
[21]	Yes	No	No	No
Proposed system	Yes	Yes	Yes	Yes

3. PROBLEM STATEMENT

A. System model and threat model

The proposed auditing system comprises two modules with a specific connection between them, the first one is the server of cloud, which is owned by the CSP who has the experience and infrastructure to serve the cloud storage, also, to provide effective services to its users, to produce, update and demand data recovery. The second one is the Client who is responsible for the personal data or the organization's data, as he sends the data to the cloud with his responsibility for its integrity and confidentiality at the same time.

We assume having a reliable cloud service provider that most of the time behaves correctly. However, the CSP may remove data that is rarely retrieved. Or the cloud provider may mask corruption of data to preserve its reputation. Therefore, the data integrity used in cloud computing is connected to two types of threats. The first threat is the integrity threat, meaning that the attacker will observe the data and then work to provide forensic evidence of fraud. While the second threat is called a privacy threat, which is that the attacker tries to obtain additional information by monitoring data, such as the content or type of data.

System goals

- The proposed system is aimed to do data privacy preservation, security of data and lightweight operations.
- Data integrity guaranty: Proving the integrity of data stored in the cloud and the security of the proposed system.
- Privacy preservation: The proposed system must meet data privacy during the data integrity process.
- Lightweight operations: The proposed system should reduce the computation operations for cloud computing users to achieve efficiency.
- Data dynamic operations: To ensure that data is stored dynamically, even if cloud computing users update their data such as delete or insert some data in the cloud computing.
- Confidentiality of data: To ensure the security of the proposed system from an untrusted cloud service provider or any external attacker.

B. Security paradigm

The security paradigm of the suggested system is designed to protect the data integrity verification process. The verification scheme is said to be more secure:

1. 1. If no polynomial algorithm exists, it can pass validation with the lowest possible probability.
2. If a polynomial extractor can retrieve the original data by conducting challenge and response operations.

The model of integrity protection lets adversary query data. The adversary may be a dishonest (CSP) who interacts with a challenger (user). The integrity game involves the following steps:

1. The Setup phase is run by a challenger to generate its private and public keys. The public key is sent to the adversary while the private is saved secretly.
2. The number of polynomials for queries of Oracle is executed by the adversary. The adversary can do a hash query, and sign query on the oracle model.
3. Finally, the adversary creates a forged proof that corresponds to the file block. The adversary wins the game if and only if, the signature is a valid signature and the data block is never requested through hash and signature queries.

With previous implementations, if the adversary successfully wins the game, we can create a simulator

that can fix CDH problems. So, the security of our system is based on assumptions of CDH stiffness on the bilinear mappings in the Oracle paradigm.

C. Preliminaries

The proposed system is constructed on the bilinear maps with the addition of the BLS signature; these two concepts will be discussed.

1)Bilinear maps

Let G_1, G_2 and G_T are multiplicative cyclic groups of prime order p. Let g_1, g_2 be the generator of G_1, G_2. A map e: $G_1 \times G_2 \rightarrow G_T$ will be a bilinear map if it achieves the following features [16]:

- Calculable: efficient process happens in computing the map (e).
- Bilinear: $e(x^a, y^b) = e(x, y)^{ab}$ for all $x, y \in G_1, G_2$, whereas $a, b \in Z_p$;
- Non-degeneracy: $e(g_1, g_2)$ guarantees not equal to 1.

2)Boneh-Lynn-Shacham (BLS)signature

It is a short digital signature that allows the user to verify the authenticity of the signer. And is proven secure in the random oracle model it was presented in [17]. It has been used because the short signatures are needed in cloud environments where there is a strong requirement that minimum bandwidth is used. Where the BLS produces short signatures compared to other signatures used such as RSA and DSA. For instance, when uses a 1024-bit modulus, RSA signatures are 1024 bits long. Likewise, when uses a 1024-bit modulus, standard DSA signatures are 320 bits long. Elliptic curve variants of DSA, such as ECDSA, are also 320 bits long. A 320-bit signature is too long to be keyed in by a human. While the BLS signature scheme whose length is about 160 bits and which provides a level of security similar to that of 320-bit DSA signatures. Signature generation is relatively fast, signature verification is slightly less computationally complex as it includes a computationally unexpansive pairing operation. Signature aggregation is one of the main features of BLS signing, as it is possible to aggregate multiple signatures into multiple messages using multiple public keys into a single signature. A gap is a group whose Diffie-Hellman computation solution is very hard, but this issue can be solved well, because of its properties including non-degraded, efficiently computable, and bilinear pairings [17].

Assume $G_1 \times G_2 \rightarrow G_T$ be non-degraded, calculable and bilinear whereas G_1, G_T are sets of prime order p. Let g be a creator of G. Assume a case of the computational Diffie Hellman problem [18], (g, g^a, g^b). The pairing function e does not help us to compute g^{ab}, the resolve to the computational Diffie–Hellman. Therefore, estimates indicate that this situation is difficult to solve. Assumed g^c, maybe we'll check to see if $g^c = g^{ab}$, this is done without knowing the a, b and c values, by proving whether $e(g^a, g^b) = e(g, g^c)$ true. By making use of the bilinear property $a + b + c$, therefore we appreciate it if $e(g^a, g^b) = e(g, g)^{ab} = e(g, g)^c = e(g, g^c)$ then, due to G_T is a prime order set, then $ab = c$. The key distinguishing feature of the BLS signature is the grouping of several signatures into several messages utilizing several keys to be able to join into one signature. The BLS have three functions: The key generator, The signing, and the Verification.

Key generator: It selects a random value $a \in Z_p$ in the interval $[0, r-1]$ which denotes the private key. The owner publishes the g^a which denotes the public key.

1. **Signing:** It takes the value a, and message m, then computes the signature by hashing the message m, as $h = H(m)$. The product of the Signing function is the signature that is $S = h^a \in G_1$.
2. **Verification:** This function takes S and g^a to prove that $e(S, g) = e(H(m), g^a)$.

4. THE PROPOSED SYSTEM

A. The auditing system based on BLS signature

The suggested data integrity auditing scheme involves six algorithms that are performed in two phases, that is the initial phase and the verification phase. The user directly sends the blocks to the CSP. The file is seen as a collection of encrypted blocks $\{e_1, e_2, \dots, e_n\}$, with the identity of each encrypted block as id_i. The user may wish to use the data file E in the future. The main problem is making sure that the cloud data file must remain intact. So the cloud computing user performs an integrity audit of the data.

Initial phase: It includes three algorithms which contain D_protection, K_generator, and S_generator. The D_protection is responsible for splitting and encrypting the data. The K_generator algorithm is primarily responsible for generating the keys, while the S_generator algorithm is responsible for generating signatures for data integrity auditing.

D_protection (F) $\rightarrow (e_n)$ In the D_protection, the user splits the file (F) into a set of blocks $(b_1, \dots, b_n)$, depending on the number of rows (r) as $F = (\{b_1, b_2, \dots, b_n\}, r)$. To achieve the confidentiality of the data stored in the cloud, the blocks are encrypted by the AES algorithm. With this, we will get the encrypted data file with the name $E = (e_1, \dots, e_n)$.

K_generator (k) $\rightarrow (y, x)$: In the K_ generator, the user produces the public key (y) and (x) as a secret key User randomly takes the parameter k as input to produces the key $x \in Z_p$, after that generates $y = {g_2}^x \in G_2$ as a public key.

S_generator $(y, x, R, E) \rightarrow S_i$: This algorithm is performed by the cloud user to generate the signatures. The input of the S_generator algorithm is x and y as private and public keys respectively, as well as r_1, r_2 as a random value $\in Z_p$, and the encrypted file E. The output of this function is the signature $S_i \in G_1$, where S_i is the single authentication identifier of each block for the file E. The signature represented from this algorithm is computed by Equation (1).

$$S_i = (H(id)^{R_i})x_i, \quad (1)$$

here, the (id) is denoted for the block index and H is the hash function to the block index $H(id) \in G_1$. The set of signatures on E appeared as $\{S_i\}_{[1\leq i\leq n]}$, with equivalent $x_i \in Zp$, $y_i = {g_2}^{x_i} \in G_2$, and the $R_i = r_1 + r_2$, where $R_i \in Z_p$ and the user creates metadata (M_d) consisting of $(A_i, B_i$, Block index $(id_i))$, where $A_i = H(id)^{r_1}, B_i = H(id)^{r_2}$ is created by the user for each block of the file E. Then he sends the (E, S, y) to cloud computing and deletes E and S from its local storage.

Verification phase: The Verification stage also includes three algorithms which include Ch_generator, Response, Check Proof. Once the cloud user needs to check the data, he sends a request to CSP to set up the audit process by using the Ch_generator algorithm.

Ch_generator $(M_d) \rightarrow chall$ *:* The input of this algorithm is the metadata (M_d), and it will generate a challenge (chall), and send it to cloud computing. Define $(chall) = \{i, q_i\}$, where $i = 1, \ldots k$ denotes the challenge set produced by random values q_i , one for each block. The CSP sends a query about the audit process to the user to agree to that query, if this is correct, the user sends the approval to the provider, and the provider creates proof (P) of integrity matching the received challenge (chall) using the Response algorithm.

Response $(y, chall, S, E) \rightarrow P$: This algorithm run by CSP to produce the proof (P) of cloud data integrity, the input is $(y, chall, S, E)$, and the final output is $P = (S, y)$, the S is the aggregate signatures of the challenger blocks (k) and is presented below:

$$S = \prod_{i=1}^{k} S_i, \text{where } 1 \leq i \leq k \quad (2)$$

To audit cloud data, the user performs this process through the Check Proof algorithm:

Check Proof $(y, P, chall, M_d) \rightarrow (yes\ or\ No)$ *:* The Check Proof algorithm is used by the user to check the returned proof (P) from the cloud. The inputs of this algorithm are $(y, P, chall, M_d)$ and the output is yes or No. This algorithm extracts the indexes of the challenged blocks, where index $= id_i$ and proves the next equation:

$$e(S, g) = e\left(\prod_{i=1}^{k}(A_i . B_i, y_i)\right), \quad (3)$$

If Equation (3) checks the proof, it returns Yes, otherwise it returns No.

B. Support data dynamic

In a cloud computing environment, cloud data can be accessed, in addition to the data being frequently updated by cloud users [19]. Hence, support the data dynamically is also important. We now show how the system was able to perform the data integrity verification process when making any modification to the data stored in the cloud such as inserting, deleting or updating any data block and thus enabling the system to deal with dynamic data.

1. *Insert data block:* Data insertion process denotes introducing a new user-defined block afterwards the indicated location in the encrypted user file (E) stored in the cloud. Suppose he needs to addition block (e) after the block (k^{th}). In the beginning, the user chooses the block he wants to insert from his local files and encrypts it, then produces the private key (x), the public key (y), and chooses the random value (R) to create the (S) with the (M_d), to this block. Finally, the user loads the new block e_i with its associated values (S_i, y_i) to the cloud to be stored in its own data file E.

2. *Data block modification:* With this process, users can update and modify their cloud data. It denotes the process of replacing the indicated data blocks with new blocks. If the cloud customer requests to modify the block (e_i), the customer will send a request to CSP to download the block (e_i) to make the necessary adjustments to it, then encrypt this updated block and with the K_generator algorithm it will produce the public and private keys and select a new value (R) for this block, to the new (M_d) and the signature (S_i) of the updated block$(e`_i)$ is created by using equation (1). Then, it loads the modified block $(e`_i)$ with its dependent value $(S`_i, y`_i)$ to store in the cloud, after the user sends a request to the cloud to delete the old block (e_i) and store the new block $(e`_i)$ instead.

3. *Deleting a block of data:* This process refers to deleting a definite block of user data, and then all following blocks are moved forward in the user's file. The process will be as follows if the customer wishes to delete the (e_i), then he will send a request

to CSP that contains the index (id_i) of the block to be deleted (b_i), so the provider will search for this block and delete it with the associated values (S_i, y_i) and thus the block (e_i) is deleted after sending confirmation of deletion by the CSP.

5. EVALUATION

To evaluate the suggested scheme, we evaluate the analysis of security and performance analysis in this section.

A. Security analysis

Initially, we analyse the security of the suggested scheme, with system correctness, data integrity protection privacy-preserving and confidentiality, and the Unpredictability of tokens.

1)Correctness

We establish the correctness of the suggested system by providing the encrypted data file $E = (e_1, \dots, e_n)$ besides signature $S = (S_1, \dots, S_n)$ on file E, the n is the number of encrypted blocks.

To validate our proposed system, we need to validate Equation (3) by the characteristics of the bilinear mappings. The user chooses random indexes (id_i) for file E with corresponding A_i, B_i then the user calculates the challenge using the Ch_generator algorithm with the challenge chall, the CSP calculates response using the Response algorithm with proof $P = (S, y)$. Finally, the user verifies the correctness of the response proof (P) by proving Equation (3) as follows:

$$
\begin{aligned}
e(S, g) &= e\Big(\prod_{i=1}^{k} (A_i . B_i), y_i\Big) \\
&= e\Big(\prod_{i=1}^{k} (H(id_i)^{r_1} . H(id_i)^{r_2}, g^{x_i}\Big) \\
&= e\Big(\prod_{i=1}^{k} (H(id_i)^{R_i}, g)^{x_i} \\
&= e\left(\prod_{i=1}^{k} (H(id_i)^{R_i})^{x_i}, g\right) \\
&= e(S, g)
\end{aligned}
$$

Consequently, the suggested system can correctly verify data integrity.

2) Ensure data integrity is protected

In this subsection, the proposed system security adversarial model is presented. The security of the proposed system against forgery is created under the selective block of data attack in a random Oracle model by simulating a game between adversary and challenger. We will demonstrate that if the adversary can produce forgery with a nonnegligible probability of success on signing the data block, then algorithm $\mathcal{B}$ can generate a solution to the CDH problem. In the game, the adversary can access the following three Oracle queries:

1. System parameter query: The adversary may request a set of system parameters from the challenger (y, g), As a result, the challenger returns the parameters to the adversary.
2. Hash Query: The adversary may request queries of hash at any time, and a challenger maintains a list to respond to that query.
3. Sign Query: The adversary may request queries of signature, and a challenger replies to this query with a signatureS_i.

After several queries, the adversary can generate a proof P^*, and he wins the game if and only if the S^* is considered to be a true signature on the data block (e_i), and the adversary did not issue any Sign query for block (e_i). Therefore, the security of the proposed system is denoted as follows:

Definition 1. The proposed system is unforgeable in a selected data block attack type if and only if the probability of success of a probabilistic polynomial-time adversary is negligible.

Assuming CDH problem hardness, the security of the suggested scheme is shown hereby:

Theorem1. If the adversary has an advantage ε to create a forgery in time t on the block e_i by simulating the game and creating the queries to Hash Query, and qs queries to Sign Query, then a CDH problem can be resolved through probability: $\varepsilon' \geq (1/(q+1)e)\,\varepsilon$ in time: $t_B \geq t + t_{ex}(qh + qs)$. The *e* represents the base of the natural logarithm.

Proof. Let the adversary be an attacker interacts with the challenger. The challenger using algorithm $\mathcal{B}$ to solve the CDH problem through probability ε' in time t_B by providing a sample (g, g^a, g^b) of a CDH problem with G, prime order p, and the $a, b \in Z_p$ as inputs. To compute the $g^{ab} \in G$. The adversary creates queries at game stages, and to respond to these queries the algorithm $\mathcal{B}$ is implemented by the challenger as follows:

Parameter Query: Initially, the adversary requests algorithm $\mathcal{B}$ to set up the system. Algorithm $\mathcal{B}$ choices $x \in Z_p$ as a secret key, and generates its public key as $g^x \in G$ and, then algorithm $\mathcal{B}$ sends (g, g^x, G) to the adversary, and saves x as the secret.

Hash Query: The adversary can create a Hash Query at any time. To keep track of each block e_i ever queried by the adversary, algorithm $\mathcal{B}$ creates a list L: $\{id_i, m_i, t_i, c_i\}$, which is initially empty. To answer this query the algorithm $\mathcal{B}$ behaves as follows:

(1) Algorithm $\mathcal{B}$ checks L for any previous query on [id], if found then $\mathcal{B}$ responds $H(id_i) = m_i$ as the answer to the adversary.
(2) Otherwise:
(a) The algorithm $\mathcal{B}$ flips a coin c where $c_i \in \{0,1\}$.

- If $c_i = 0$, then the probability $(pr = \delta)$, The algorithm $\mathcal{B}$ selects a random $t \in Z_p$, b^{1-c_i} and calculates $m = g^{t.b} \in G$.
- If $c_i = 1$ then the probability $(pr = 1 - \delta)$, and algorithm $\mathcal{B}$ computes m become $m = g^t \in G$.

(b) After that the algorithm $\mathcal{B}$ adds the new values to L and responds to $H(id_i) = m_i$ as an answer to the adversary.

Sign Query: The adversary can create a Sign Query for a block (id) and algorithm $\mathcal{B}$ replies to this query as follows:

(a) Initially, algorithm $\mathcal{B}$ issues a hash query and checks the value of c_i.
(b) If $c_i = 0$, then algorithm $\mathcal{B}$ reports failure and stops.
(c) Else, if $c_i = 1$, then $m_i = g^t$ and calculates the signature $S_i = g^{t.x} = m^{R.x}$.
(d) Algorithm $\mathcal{B}$ responds to an adversary with S_i.

Forgery: Finally, the adversary generates a forged signature S^* for a block $[id_f]$ such that no Sign Query has been issued for $[id_f]$ to generate a proof P^*.
Now the algorithm $\mathcal{B}$ does the following steps to produce a forgery:

(a) Algorithm $\mathcal{B}$ issues a Hash Query for $[id_f]$ and checks the value of c_i,
(b) If $c_i = 1$, then the algorithm $\mathcal{B}$ stops.
(c) If $c_i = 0$, then $H(id_i) = m_i = g^{t.b}$, if the $P^* = (S^*, y^*)$ is valid proof then we can get $S^* = (g^{t.b})^{R.x}$, and $g^{b.x} = S^{*1/t.R}$. The algorithm $\mathcal{B}$ recognizes the values of (S^*, t, R), therefore algorithm $\mathcal{B}$ can calculate the $g^{b.x}$ or $\mathcal{B}$ solves the CDH problem.

Additionally, the next proof demonstrates that the problem of CDH can be resolved by algorithm $\mathcal{B}$ through probability: $\varepsilon' \geq ((1/qs + 1)\,e)\varepsilon$. Therefore, all three cases must be true for algorithm $\mathcal{B}$ to succeed:
C 1: Algorithm $\mathcal{B}$ will not stop due to any query adversary for signatures.
C 2: The adversary generates a valid forged signature S^* on the block $[id_f]$.
C3: Case C_2 happens and $c = 0$ for a group having $[id_f]$ in the list.
The probability of these three cases happens is as $pr[C1 \cap C2 \cap C3] = Pr[C1].Pr[C2|C1].Pr[C3|C1 \cap C2]$.
The following claims afford a minimum for each of the preceding cases.

Claim1: The probability that $\mathcal{B}$ will not be stopped due to at least any adversary query for a signature is $(1 - \delta)^{qs}$.

$pr[C1] \geq (1 - \delta)^{qs}$

Claim2: The probability that $\mathcal{B}$ will not be stopped due to any query adversary for signatures and the adversary generates a valid forged signature S^* on the block $[id_f]$ is: $Pr[C2|C1] \geq \varepsilon$.

Claim3: The probability that $\mathcal{B}$ will not end after the adversary has produced valid forgery at least is: $Pr[C3|C1 \cap C2] \geq \delta$.

Proof: To proof the claims presented above, therefore in *Claim1*, when the $pr[c_i = 1] = (1 - \delta)$ then the probability that algorithm $\mathcal{B}$ does not stop is $(1 - \delta)$. Since it takes approximately Sign Query q_s, so the probability of algorithm $\mathcal{B}$ not stopping after queries by the adversary is at least $(1 - \delta)^{qs}$.

Also in *Claim2*, the responses to Hash Query are as in the real attack since each response is uniformly and independently distributed in G. All responses to Sign Query are valid. Therefore, the adversary will produce a valid forged signature, then the probability at least ε. Moreover, in *Claim3*, the C1 and C2 states happen, Algorithm $\mathcal{B}$ will not work only if the adversary can configure a forgery with $c_i = 1$ in the L. If the adversary does not request the signature, algorithm $\mathcal{B}$ provides $H(id_i)$ to him based on the c_i. Meanwhile the adversary will not be able to issue a query about the signature, this means the value of c_i, will not be known by the adversary and thus Probability at least δ.
After we used the bounds of the above claims in:

$$pr[C1 \cap C2 \cap C3] = Pr[C1].Pr[C2|C1].Pr[C3|C1 \cap C2]$$

Therefore, we get the following Equation.

$$\varepsilon' = ((1 - \delta)^{qs}.\varepsilon.\delta)$$
$$\varepsilon' = (\delta(1 - \delta)^{qs}.\varepsilon) \qquad (4)$$

By using the term differential of Equation (4), we get the smallest value of ε' as follows:

$$d\varepsilon'/d\delta = d(\delta(1 - \delta)^{qs}.\varepsilon/d\delta$$
$$= (1.(1 - \delta)^{qs}.\varepsilon) + (\delta.qs\,(1 - \delta)^{(qs-1)}.(-1))\varepsilon$$
$$= (1 - \delta)^{(qs-1)}\varepsilon((1 - \delta) - \delta.qs)$$
$$= (1 - \delta)^{(qs-1)}\varepsilon(1 - \delta(1 + qs))$$

Substitute $d\varepsilon'/d\delta = 0$,

$(1 - \delta)^{(qs-1)}\varepsilon(1 - \delta(1 + qs)) = 0$, then (δ) pick out $(1/(1 + qs))$, after replace δ in Equation (4), we get $\varepsilon' \geq 1/(qs + 1)[1 - 1/(qs + 1)]^{qs}.\varepsilon$.

For a large value of qs, the probability of adversary signature queries is at least $1/e$ instead of $[1 - 1/(qs + 1)]$, therefore the CDH problem can be resolved by $\mathcal{B}$ with probability $\varepsilon' \geq [(1/qs + 1).e]\varepsilon$.

The runtime of algorithm $\mathcal{B}$ is the same as the adversary time in addition to the time taken to reply to

hash and signature queries. Every query involves an exponentiation process that represents t_{ex} time. Therefore, the whole runtime is at most $t_B \geq t + t_{ex}(qh + qs)$ as necessary. This ends Proof of Theorem 1.

3)Privacy-preserving and confidentiality

When the user generates the signature, he creates (R) which consists of the sum of two different random values (r_1, r_2). Thus, the user will verify the validity of the signature retrieved from the cloud through the values of (r_1, r_2) using Equation (3). Therefore, from the response of the cloud computing, the response is blinded by the value of (R), and because of the strength of the DL problem, the information about the (R) cannot be known and cannot be leaked to the server.

Additional, the confidentiality of cloud computing user data from both the cloud service provider as well as from the cloud computing users must be maintained. This is why security remains one of the biggest concerns in a cloud environment. The proposed system has provided a strong solution to one of the most common problems facing cloud storage, which is the issue of privacy and confidentiality, by working on the principle of encryption, thus ensuring the confidentiality of cloud data. But we must have encryption technology that is sensitive to plain text and keys also. At initial, we know that the AES key requires the similar length as the plaintext and its explanations with which this key can resist any brute force attack. This is because the correct and incorrect keys have only one difference. Thus, the results revealed that at the time of decryption, if the wrong key was used, which may differ slightly from the correct key, a significant difference was found between the original file and the encrypted file.

Therefore, utilizing the incorrect information cannot recover the original data, thus leading to a failure to decrypt the data. Therefore, we believe that the AES algorithm is very sensitive to plain text and thus guaranteeing the security of the proposed system and realizes its goals.

B. Performance analysis

The implementation of the suggested system has been done in the python language through an Intel (R) Core i5 CPU at 3.60 GHz and 8.00 GB RAM. We go into detail to show the efficiency and security of the suggested system. The efficiency concept means that the suggested auditing system provides an assurance of data integrity with reduced computational and communication overhead. To test the proposed system performance, the Berka dataset [20] is used. To implement the suggested scheme, we depend on the dataset for the original data file, therefore the customer divides the data file into blocks (100 to 500 blocks) thus that the block size is (1 KB). Moreover, we use the Dropbox cloud method to deploy our data and to check the suggested scheme. In the next subsections, we will now evaluate the performance of the suggested system according to the cost of the Computation and the cost of communication as follows:

1)Computation cost

To evaluate the generating performance of both the keys and the signatures, we generate the keys and the signatures for a dissimilar number of blocks from 100 to 500 growing by 100 in our experiment. Such as presented in Figures 1 and 2, the time cost of generating signatures increases linearly through the increase in the number of blocks. Keys generation time ranges from 0.021 to 0.046 seconds. While the time to generate signatures ranges from 0.071 to 0.286 seconds.

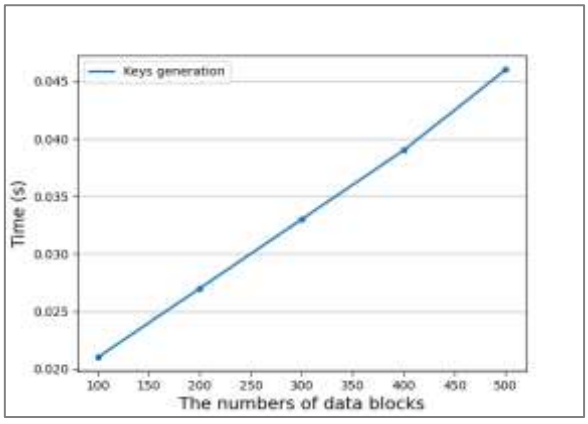

Figure 1. The computation overheads of keys generation

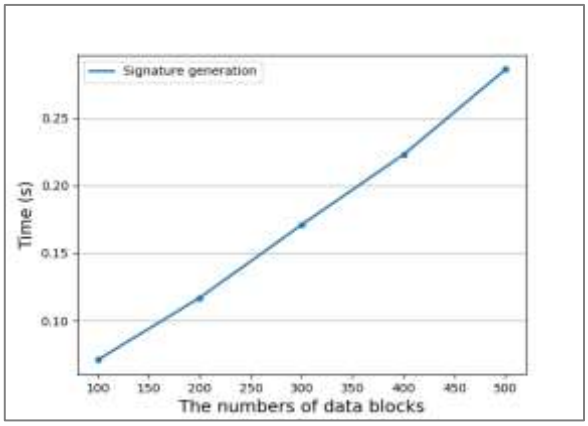

Figure 2. The computation overheads of signature generation

It can be noted from Figures 1 and 2 that the proposed system takes very little time in the keys generation process as well as in the signatures generation process, and this guarantees the efficiency of the suggested scheme.

For an integrity audit, we illustrate the computational overheads in Figures 3 and 4. In our experience, the number of data sets challenged varies from 100 to 500. As presented in Figure 3, we realize that the computational costs of challenge generation and check the proof on the

user-side grow linearly through the number of data challenges. The computational cost of challenge generation ranges from 0.031 to 0.445 seconds. Matched to check the proof time, the challenge generation time grows slowly, from 0.159 seconds to 0.833 seconds. From Figure 4, we can see that the computational cost of generating proof on the cloudy side differs from 0.411 to 1.997 seconds.

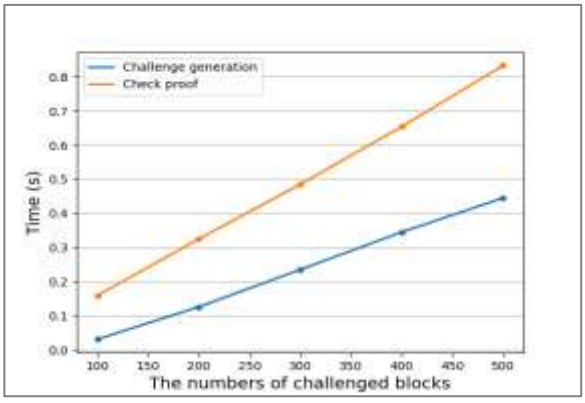

Figure 3. The computational overheads of challenge generation and check the proof

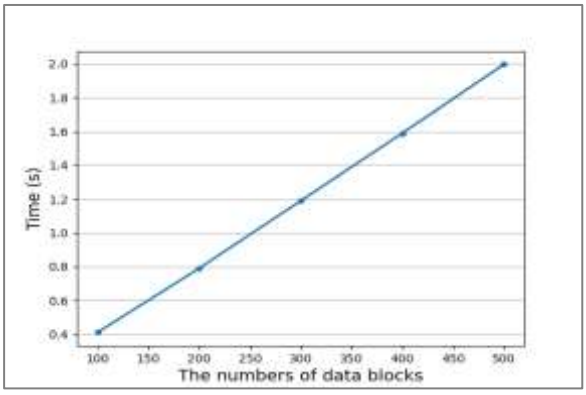

Figure 4. The computation overheads of generating proof

As we have seen in Figures 3 and 4, the computational overheads of challenge generation, proof generating and check the proof as to the number of challenges increases. This is appropriate for performance analysis because while the block number challenged is in height, the extra random values must be set in the challenge generation, plus the cloud has incremental computations while the user has additional processes. Therefore, we can achieve that additional data challenge. However, the time required for the number of data blocks that have been challenged remains very acceptable and this indicates the efficiency of the suggested system.

To evaluate the efficiency, we compare the suggested system to [15] in terms of verification side and server-side with a 1 KB data block size for 500 blocks and summarize the results in Table II. We can notice from Table II that the proposed system has a better computation cost than the system [15] so that the results show that the proposed system has a computation cost of fewer than four times. As well as for comparison with [22], it turns out that the proposed system has better efficiency.

TABLE II. COMPARISON OF DATA INTEGRITY VERIFICATION

Side	The proposed system (s)	[15] (s)	[22] (s)
Verification-side	0.80	4.5	1.5
Server-side	1.79	8.4	3.2

Table III gives a comparison of the computational overheads between the proposed system and [21,23,24] at different stages.

TABLE III. COMPARISON OF THE COMPUTATION OVERHEADS

	Proposed system	[21]	[23]	[24]
Generate Signatures	0.071	1.702	2.91	2.95
Generate Challenge	0.031	0.034	1.53	1.57
Check Proof	0.159	1.010	0.22	0.26
Generate Proof	0.411	0.802	0.32	0.35

As presented in Table III, we can realize that the proposed system and the system [21] have nearly the same computation cost in the part of challenge creation. So, the proposed system and the system [21] have equal efficiency when handling the same data. While in the proof generation, the proposed system has less computation overhead, therefore, the proposed system is better than the system [21]. As well as in the phase of proof verification. Therefore, the comparison illustrations that the proposed system has better efficiency than the system [21]. Also, in comparison with [23,24], it was found that the proposed system has high efficiency.

2)Communication cost

Communication costs are due to data sending to the cloud, data retrieval, and responding to the challenge of auditing, and these are unavoidable matters. As mentioned in subsection 4.1 of the suggested scheme, the client selects random blocks to challenge. The burden of communication is done because the audit process in the proposed system contains double transmissions. First, it will be for the challenge data file E which is represented as chall = (i, iq). The cost is based on the number of n blocks sent by the client for the audit. While the second is for the challenge and the corresponding response to the challenge in the proof P = {S, y}. Thus, the

communication cost will be O(n). Now, we will calculate the times of uploading and downloading blocks from the cloud in order to verify the effectiveness and efficiency of the proposed system. Initially, after the process of dividing the data to be stored into a set of n blocks, the resulting blocks are encrypted, so that we have encrypted blocks ready to be sent to the cloud. As such, to display the upload and download results, Figure 5 and Figure 6 illustrate this.

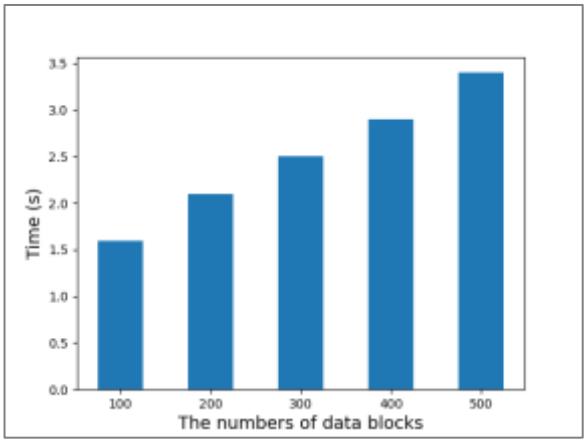

Figure 5. Upload blocks

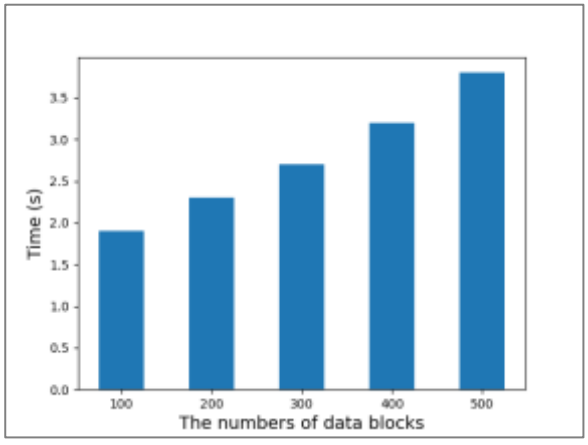

Figure 6. Download blocks

6. CONCLUSION

This paper proposed an auditing system for cloud computing data. This system used a short signature from BLS to ensure audit and maintain data privacy, and the proposed system extends to data dynamically, where the user can perform insertion, deletion, and modification of data to enhance efficiency. Moreover, the data integrity calculation cost is more low for the user auditing, because the data verification signature contains one element, thus the cost of storing and transmitting the signature can be reduced. Comprehensive analysis shows that the suggested scheme is very efficient and more secure. Future works could be expanded to support auditing in multi-cloud environments.

REFERENCES

[1] Vamsi, S., Raviteja, R., Selvan, M. P., & Gladence, M. (2021). Enabling Ternary Hash Tree-Based Integrity Verification for Secure Cloud Data Storage. In Advances in Electronics, Communication and Computing (pp. 447-453). Springer, Singapore.

[2] Gahlot, D., Tejasvee, S., Ranga, K. B., & Vyas, R. R. (2021). Data Security & Future Issues for Cloud Computing. In Advances in Information Communication Technology and Computing (pp. 313-318). Springer, Singapore.

[3] Ghallab, A., Saif, M. H., & Mohsen, A. (2021). Data Integrity and Security in Distributed Cloud Computing—A Review. In Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications (pp. 767-784). Springer, Singapore.

[4] Mahmood, G. S., Huang, D. J., & Jaleel, B. A. (2019). A secure cloud computing system by using encryption and access control model. Journal of Information Processing Systems, 15(3), 538-549.

[5] Saleh, W. M., Al-Ta'I, Z., T., Mahmood, G. S. (2020). Secure Distributed Data using Multi-Cloud. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 9(4), 2998-3003.

[6] Wang, C., Chow, S. S., Wang, Q., Ren, K., & Lou, W. (2011). Privacy-preserving public auditing for secure cloud storage. IEEE transactions on computers, 62(2), 362-375.

[7] Erway, C. C., Küpçü, A., Papamanthou, C., & Tamassia, R. (2015). Dynamic provable data possession. ACM Transactions on Information and System Security (TISSEC), 17(4), 1-29.

[8] Ge, X., Yu, J., Zhang, H., Hu, C., Li, Z., Qin, Z., & Hao, R. (2019). Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification. IEEE Transactions on Dependable and secure computing.

[9] Lu, X., Pan, Z., & Xian, H. (2020). An efficient and secure data sharing scheme for mobile devices in cloud computing. Journal of Cloud Computing, 9(1), 1-13.

[10] Wang, F., Xu, L., Wang, H., & Chen, Z. (2018). Identity-based non-repudiable dynamic provable data possession in cloud storage. Computers & Electrical Engineering, 69, 521-533.

[11] Sun, L., Xu, C., Zhang, Y., & Chen, K. (2019). An efficient iO-based data integrity verification scheme for cloud storage. Information Sciences, 62(059101), 1-059101.

[12] Ping, Y., Zhan, Y., Lu, K., & Wang, B. (2020). Public Data Integrity Verification Scheme for Secure Cloud Storage. Information, 11(9), 409.

[13] Shang, T., Zhang, F., Chen, X., Liu, J., & Lu, X. (2019). Identity-Based Dynamic Data Auditing for Big Data Storage. IEEE Transactions on Big Data.

[14] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., & Song, D. (2007, October). Provable data possession at untrusted stores. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 598-609).

[15] Thokchom, S., & Saikia, D. K. (2019). Privacy Preserving and Public Auditable Integrity Checking on Dynamic Cloud Data. IJ Network Security, 21(2), 221-229.

[16] Li, S., Liu, J., Yang, G., & Han, J. (2020). A Blockchain-Based Public Auditing Scheme for Cloud Storage Environment without Trusted Auditors. Wireless Communications and Mobile Computing, 2020.

[17] Boneh, D., Lynn, B., & Shacham, H. (2004). Short signatures from the Weil pairing. Journal of cryptology, 17(4), 297-319.

[18] Kumar, K. A., Babu, R. M. H., Kalaivanan, S., & Kanimozhi, V. (2020). Multiauditing Based Cloud Storage Using a Dynamic Hash Table.

[19] Fan, Y., Lin, X., Tan, G., Zhang, Y., Dong, W., & Lei, J. (2019). One secure data integrity verification scheme for cloud storage. Future Generation Computer Systems, 96, 376-385.

[20] Phil, A., Marguerite, D., & Priya, K. (2002). Credit Card Analysis of Czech Bank. https://webpages.uncc.edu/mirsad/itcs6265/group1/domain.html.

[21] Shen, W., Qin, J., Yu, J., Hao, R., & Hu, J. (2018). Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Transactions on Information Forensics and Security, 14(2), 331-346.

[22] Wang, Q., Wang, C., Li, J., Ren, K., Lou, W., 2009. Enabling public verifiability and data dynamics for storage security in cloud computing. In: European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg, pp. 355–370

[23] Xu, Z., Wu, L., Khan, M. K., Choo, K. K. R., & He, D. (2017). A secure and efficient public auditing scheme using RSA algorithm for cloud storage. The Journal of Supercomputing, 73(12), 5285-5309.

[24] Yu, Y., Xue, L., Au, M. H., Susilo, W., Ni, J., Zhang, Y., ... & Shen, J. (2016). Cloud data integrity checking with an identity-based auditing mechanism from RSA. Future Generation Computer Systems, 62, 85-91.

Ghassan Sabeeh Mahmood received the B.Sc. degree in computer science from University of Baghdad, Iraq in 2002, Iraq and the M.Sc. degree in School of Information Science and Engineering from Central South University, China in 2015. His research interests include security of cloud computing.

Noor Hasan Hassoon received her bachelor's in computer science from Diyala University, Iraq in 200, and master's in information technology form Universiti Tenaga Nasional. Her research interests include LMS, Cloud Computing and AI in education. she is currently lecturer at Department of computer science, College of Education for Pure Science, Diyala University.

Hazim Noman Abed received his bachelor's in computer science from Diyala University, Iraq in 2007, and master's in information technology from Universiti Tenga Nasional (UNITEN). His major filed of research are semantic web, Ontology, Knowledge representation, and cloud computing. He is currently lecturer at Diyala University, College of science, and Department of computer science

Baidaa Abdulrahman Jaleel received a B.Sc. degree in computer science from the University of Diyala, Iraq in 2007, Iraq, and the M.Sc. degree in computer science from the University of Diyala, Iraq in 2021. Her research interests include cloud computing.