

Remote Data Integrity Verification System for Cloud Computing Based on Efficient Signature

Bassam Hindy Hameed^{1*}, Ghassan Sabeeh Mahmood¹, Hazim Noman Abed¹, Mohammed Ahmed Salman²

¹University of Diyala, College of Science, Computer Science Department, Diyala, Iraq

²Imam Aladham University College, Baghdad, Iraq

ABSTRACT: Cloud storage services allow remote data storage, freeing up local space and offering resource flexibility. To maintain data integrity, various schemes have been introduced. However, existing remote data integrity verification schemes using RSA and BLS (Boneh, Lynn, and Shacham) signature mechanisms have limitations. BLS relies on a probabilistic and inefficient hash function, leading to computational complexity, while RSA incurs significant overhead. This study presents an innovative approach for cloud data integrity verification, which utilizes a novel short signature technique that relies on bilinear pairings. Our scheme employs efficient cryptographic hash functions like SHA-1 or MD5, unlike BLS. It requires fewer pairing operations than BLS and RSA, resulting in a lower computational cost. Additionally, our scheme ensures public auditability and data privacy, with theoretical analysis confirming correctness and security properties.

KEYWORDS - Data integrity, Cloud computing, Cloud Service Provider, Third-party auditor.

I. INTRODUCTION

The emergence for very promising paradigm in the field of information technology has emerged: cloud computing. This paradigm has brought about a significant shift in the way data storage and administration are approached, since it involves a transfer from traditional local solutions to cloud-based alternatives. As the volume of user data continues to grow exponentially, it is crucial to develop practical and efficient methods for storing and managing data while upholding security and usability standards, particularly in the context of cloud storage solutions [1]. Users, on the other hand, relinquish physical possession of their data when they upload it to external cloud services. To cut administrative and storage expenses, certain cloud service providers (CSPs) may erase less frequently accessed data. Additionally, system failures and operational mistakes might result in data loss. [1], or malfunctioning hardware and software on the cloud servers. Several data integrity-checking approaches Efforts have been undertaken in recent years to establish the veracity of cloud data without the need to fully retrieve the outsourced files. Many of these plans are dependent on the BLS [2][3] and RSA signature mechanisms [4]. However, it has become evident that these existing schemes face certain challenges. For instance, despite offering strong privacy protection, the BLS system uses the Map to Point acceptable encoding function, which is also utilized by traditional cryptographic pairings schemes. In addition, the BLS has a poor efficiency when it comes to the computing of batch signatures in the context of situations including massive data. Furthermore, BLS entails computational complexity. On the other hand, RSA (referred to as ID-SEPA) utilizes random masking techniques to safeguard user data privacy but incurs high computational cost. In order to address the aforementioned challenges pertaining to computational overhead and signature efficiency in the aforementioned signature mechanisms, this research paper presents a viable integrity verification scheme for cloud data. The proposed scheme is founded on a novel short signature scheme derived from the Computational Diffie-Hellman Problem (CDHP), which leverages bilinear pairings [5]. It is imperative to acknowledge that inside pairing-based cryptosystems, the computation of pairings stands out as the operation that consumes the

greatest time. Despite efforts to analyze pairing complexity and accelerate the computation process [6], pairing computations still remain time-consuming. The strategy we present aims to reduce the number of pairing operations needed in comparison to the BLS scheme. Moreover, it boasts a lower computational cost than both BLS and RSA, making it a more efficient solution. Additionally, our scheme provides robust privacy protection.

Contributions. Our work's contributions can be summarized as follows:

- (i) The present study introduces a novel approach to address the issue of remote data integrity verification in cloud computing. This approach is centered on the utilization of a newly developed short signature, which effectively eliminates the need for users to incur additional costs associated with conducting audits. Users can conduct public audits of their data through a TPA.
- (ii) Our system ensures privacy preservation and supports dynamic data operations.
- (iii) The proposed system ensures data protection in cloud storage with data partitioning, encryption using AES algorithm.
- (iv) In the process of signature generation, have been successful in reducing the computational weight of the hash function, which has led to an increase in the effectiveness of our method in contrast to that of BLS and RSA.
- (v) Additionally, our system resists adaptive chosen-message attacks and provides high security under the CDH problem assumption in the random oracle model.

Organization of the article: the remaining sections of this article will be organized as described below. The connected work is discussed in Section 2. The preliminary notions are introduced in Section 3. The system model and the Security Risk Assessment are both discussed in Section 4, which provides an overview of both topics. In Section 5, the suggested system and algorithms are broken down in detail. Section 6 offers a thorough evaluation, including security and performance analysis with experimental results. Finally, Section 7 provides the study's conclusions.

II. RELATED WORKS

In recent years, several proposals put up to tackle the issues around integrity and privacy in cloud data. Juels and Kaliski [7] introduced the Proofs of Retrievability (POR) scheme for data possession and retrievability, but it lacks support for public auditing and has limited query capabilities. Ateniese et al. [8] introduced Provable Data Possession (PDP), presenting secure PDP schemes with integrity checks on random blocks. However, this scheme is limited to static data and incurs storage and communication overhead. Shah et al. [9] introduced message authentication code-based PDP, but his approach suffers from limitations on the number of verifications and storage overhead. In 2015, Wang et al. [10] The author initially introduced a theoretical framework for identity-based distributed proven data possession within the context of multi-cloud storage. However, subsequent analysis revealed that this framework is susceptible to proof forging attacks. Chen [11] created a remote data availability verification protocol that is based on algebraic signatures and is efficient, with support for an infinite number of verifications. However, it requires frequent challenge-update checks, leading to additional communication and computational overhead. Sun et al. [12] This work focuses on the practical use of a privacy-preserving adaptive trapdoor hash authentication tree, with a primary emphasis on the implementation of public auditing methods as the topic of investigation. Ping et al. [13] and Srinivas et al. [14] introduced public systems for dynamic data auditing using a TPA. Wang et al. [15] proposed a system supporting public auditing and dynamic data processes using BLS signatures, while Garg et al. [16] emphasized privacy preservation in their system. To ensure privacy during auditing, the method must safeguard data privacy. Wang et al. [17] The researchers proposed a privacy-preserving mechanism for the secure storage of cloud data. However, the system exhibits flaws in relation to identifying message attacks. This paper proposes a new short signature scheme, based on CDHP using bilinear pairing, to enhance cloud data integrity verification. It improves efficiency, reduces computational costs, and minimizes storage overhead compared to RSA [4] and

BLS signatures [2][3]. The scheme ensures secure data integrity, privacy protection, public auditing, and data confidentiality in cloud environments.

III. THE PRELIMINARIES

• GDH Groups and The Bilinear Map

Where the multiplicative cyclic groups G_1 , G_2 , and G_T all have the same huge prime order (P), and where $|G_1| = |G_2| = |G_T|$. Let g_1, g_2 be the generators of G_1, G_2 , respectively. $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear map if it satisfies the following properties:

(i) Bilinear: $\forall u \in G_1, v \in G_2$ and $\forall a, b \in \mathbb{Z}_p, e(u^a, v^b) = e(u, v)^{ab}$ holds.

(ii) Non degenerate: $\exists u \in G_1, v \in G_2$, such that $e(u, v) \neq IG_T$, IG_T the identity element within cyclic group denoted as G_T . If an isomorphism exists: $G_2 \rightarrow G_1$, with $\psi(g_2) = g_1$, (G_1, G_2) is a Gap Diffie-Hellman (GDH) group pair. can set $G_1 = G_2 = G$ and $g_1 = g_2 = g$ and take ψ to be the identity map.

(a) Computational co-Diffie-Hellman (co-CDH) on GDH[18]. Given $g_2, g^a \in G_2$ and $h \in G_1$ as input and $a \in \mathbb{Z}_p$, then compute $h^a \in G_1$.

(b) Decision co-Diffie-Hellman (co-DDH) on GDH [18]. Given $g_2, g^a \in G_2$ and $h, h^b \in G_1$ as input, if $a = b$ the output is yes; If not, the result is no. when "yes" is the response, say that (g_2, g^a, h, h^a) is a co-Diffie-Hellman tuple. When $G_1 = G_2$, these problems reduce to a standard CDH and DDH problems. The co-DDH problem is easy to be solved but co-CDH is hard on the GDH group [19].

• The Short Signature

This concept (short signature) is constructed using a bilinear pairing that was suggested by CDHP. The proposed theory aligns with the concept of Bilinear Pairing-based Cryptography (BLS) in order to design a signature scheme that is difficult to the Computational Diffie-Hellman (CDH) problem within a given group G . Furthermore, the proposed signature system has less overhead than BLS does. an algorithm for each of the following is included in the proposed signature: ParamGen is an algorithm for the creation of parameters. KeyGen is an algorithm for the generation of keys. Sign is an algorithm for the generation of signatures, as well as a technique for verifying signatures Ver.

provided a description of the proposed system in the following manner:

1. ParamGen. Are system parameters $\{G_1, G_2, e, q, P, H\}$.
2. KeyGen. Is randomly selects $x \in \mathbb{Z} * q$, and computes $P_{pub} = x * P$. The public key is P_{pub} . The secret key is x .
3. Signing. With a secret key denoted as x and a message represented by m , calculate the value S as $S = \frac{1}{H(m_i) + x} P$. The signature is S .
4. Ver. Verify that the equation $(H(m)P + P_{pub}, S) = e(P, P)$ holds true when you are provided with a public key P_{pub} , a message m , and a signature S .

IV. OVERVIEW OF APPROACH

• The System Model

The public auditing system that is suggested is shown in Figure 1. Three entities make up the model, and there is a clear connection between them all.

- (i) Cloud users. have the ownership of their data and seek to optimize local storage and processing resources by transferring the data to the cloud.
- (ii) CSP Servers. Users have access to a substantial quantity of storage space that is made available on the CSP servers. Simultaneously, it is the responsibility of CSP servers to efficiently carry out cloud operations, including data updating, data querying, and responding to client requests.
- (iii) The concept of the Third Party Auditor (TPA). The TPA possesses a greater level of capability and expertise compared to the clients. Individuals have the option to seek assistance from a Third-Party Auditor

(TPA) in order to conduct an assessment of the reliability and accuracy of externally sourced data on their behalf.

Users can cut expenditures on data storage and upkeep by using cloud storage, but there is a risk that the cloud service provider (CSP) could engage in unethical behavior, which raises worries about the data's integrity. To address this, an integrity checking mechanism can be established. Users can authorize a TPA to conduct security auditing, minimizing computational and communication overhead. The verification procedure is managed by the Trusted Platform Agent (TPA), with only the outcome being relayed to the user.

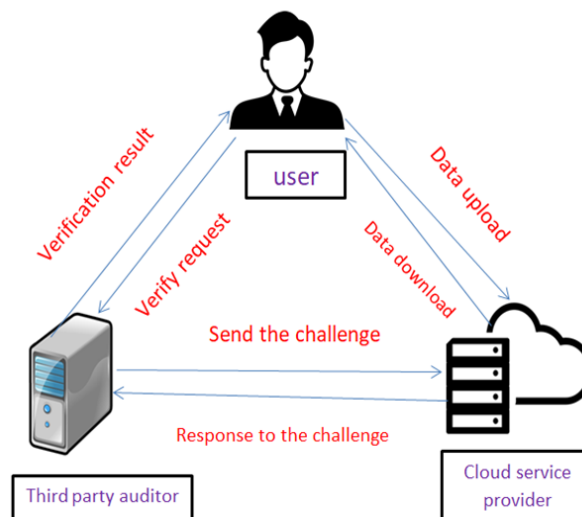


Fig. 1. The suggested system model.

• Security Risk Assessment

Assuming the CSP is dependable, there is a chance it may delete infrequently accessed files or hide data corruption. However, The Third-Party Auditor (TPA) responsible for conducting the audit is widely regarded as neutral and trustworthy, devoid of any motivation to engage in collusion with either Cloud Service Provider (CSP) or the users. Nonetheless, the TPA poses a potential risk if they can access the data after the audit. Therefore, data security within a CSP is exposed to two main threats:

1. Integrity Endangerment: Attackers can exploit data, signatures, and public keys to create convincing evidence of data fraud.
2. Privacy Endangerment: Intruders can monitor metadata and evidence to gather additional information, such as type or data content.

V. THE PROPOSED SYSTEM

Initially, the user logs in successfully and divides the data file into blocks. Each block is encrypted with AES 256 for confidentiality. These encrypted blocks, denoted as $E = e_1, \dots, e_n$, The files are transferred and stored in a remote server network commonly referred to as the cloud. The user generates both public and private keys, together with associated signatures for each encrypted block, which are subsequently communicated to the cloud. The original data file and signatures are removed from local storage. To verify data integrity, the user delegates the task to an auditor who generates a challenge. The cloud service provider receives the challenge, accepts it, and gives the auditor the proof in return. The auditor checks the accuracy of the data and informs the user of the findings.

the system parameters include cyclic additive group G_1 generated by P , cyclic multiplicative group G_2 , order represented by q , and integer ring Z_q . A

bilinear pairing $e: G_1 \times G_2 \rightarrow G_2$ is assumed to map elements from G_1 to G_2 .

The suggested system is divided into two phases: the initial phase and integrity verification phase.

- **Initial phase**

The first stage is comprised of three different algorithms: data protection, key generation, and signature generation.

Algorithm One_Protectionof Data

The user implements the Data Protection algorithm in order to safeguard the information, which consists of dividing the data file (D) into a number of separate data blocks. $(d_1, \dots, d_n)$. These data blocks are then encrypted using The Advanced Encryption Standard (AES) algorithm is utilized for the purpose of preserving the confidentiality of data, resulting in encrypted data file $E = e_1, \dots, e_n$. The algorithm consists of the following main components:

Algorithm One : Protection of Data

Input: The Data file (D).

Output: The Encrypted Data file $E = e_1, \dots, e_n$

1: Divide Data file D into multiple of data blocks $(d_1, \dots, d_n)$.

2: Utilizes an AES Algorithm to encrypt $(d_1, \dots, d_n)$ and obtain the data file $E = e_1, \dots, e_n$.

Algorithm Two_ Key Generation

Using the KeyGen algorithm, the user creates a public key (pk) and a secret key (sk).

They randomly choose a value x from a set of integers modulo q ($x \in \mathbb{Z}_q$) and calculate pk as $P_{pub} = x * P$, where P is a random value. Thus, $pk = P_{pub}$ and $sk = x$. This process is repeated for each block, with the user holding sk and publishing pk , representing an element of group G .

The KeyGen algorithm is presented through the following steps:

Algorithm Two: Key Generation

Input: These security parameter (k) $\in$ prime order (q), a random value (P) $\in \mathbb{Z}_q$.

Output: private key (x) $\in \mathbb{Z}_q$, public key (pk) $\in G$.

1. Selects a random value (x) It is a representation of the secret key that is based on the input of the security parameter k .

2. Computes the public key pk using the formula pk as $pk = x * P$.

Algorithm Three_ Signature Generation

In the signing phase, user generates a signature of encrypted data blocks using the SignatureGen algorithm. A secret key is what feeds into this algorithm as (x) its input, (P) is a random value, and an encrypted block name (m_i) from the encrypted file (E). (S_i) is the output of this algorithm is a signature, where (S_i) is the authentication identifier for each block. While (S_i) is a signatures are calculated using Eq. (1)

$$S_i = \frac{1}{H(m_i) + x_i} P_{i(1)}$$

The user employs a cryptographic hash function $H: \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ for the block name (m),

Where $|q| \geq \lambda \geq 160$. For each block in file (E), the user generates Signatures ($\{S_i\} [1 \leq i \leq n]$) on E with conforming secret keys (x_i), public keys $pk_i = x * P_i$ and a random value $P_i \in \mathbb{Z}_q$, $[1 \leq i \leq n]$. Next, the user send encrypted data blocks, digital signatures, and the public key to a cloud storage system. Descriptive data (D_{Detail}) containing (P_i ; block name (m_i)) is sent to the TPA for auditing, and the file m is deleted locally. The primary steps for the SignatureGen algorithm are as follows:

Algorithm Three: Signature Generation

Input: The secret key $(x) \in \mathbb{RZ} * q$, a random value $(P) \in \mathbb{Z}q$, encrypted data file $E = e_1, \dots, e_n$.

Output: signatures $(S_i) \in G$

1. The computation of the hash value for a block name is performed through the utilisation of the cryptographic hash function known as $SHA - 1$.
2. The signature can be generated by utilising Eq.(1).
3. Create the descriptive data $((D_{Detail}) = P_i, \text{block name}(m_i))$
4. The user Upload the encrypted data file (E) , the generated signatures (S_i) , and (pk_i) the public key to cloud storage.

• Integrity verification phase

Three algorithms are involved in the integrity verification phase: Challenge Generation, Response, and Verify Proof. Third Party auditor used by Users for auditing server-side data. The user activates TPA account to send descriptive data (D_{Detail}) for auditing. The TPA interacts with Cloud Service Provider to illustrate file F's integrity.

Algorithm Four _ Challenge Generation

The TPA generates challenges by selecting c elements from the set $\{1, 2, \dots, n\}$ to form the set $R = \{t_1, t_2, \dots, t_c\}$, where $1 \leq t_1 \leq \dots \leq t_c \leq n$. For each $i \in R$, the TPA selects a random $r_i \in \mathbb{Z}q$. The challenge message sent to the CSP is denoted as $R = \{(i, r_i)\}, t_1 \leq i \leq t_c$.

The *ChallGen* algorithm consists of the following main steps:

Algorithm Four: Challenge Generation

Input: The descriptive data $(D_{Detail}) [P_i, \text{block name}(m)]$

Output: challenge (R)

- 1: The user begins using TPA.
- 2: User transmit the D_{Detail} to the TPA for auditing.
- 3: TPA generates a challenge, $R = \{(i, r_i)\}, t_1 \leq i \leq t_c$, where (i) represents the challenge block number using random values $r_i \in \mathbb{Z}q$, one for each data block.
- 4: TPA transmits the generated challenge (R) to the cloud.

Algorithm Five _ Response

Upon receiving a challenge from the Third Party Auditor (TPA), Cloud Service Provider (CSP) generates a integrity proof that corresponds to challenge received for designated data block. The ability to combine several signatures for multiple blocks into a single signature can be achieved using Equation (2). Similarly, Equation (3) enables the aggregation of multiple public keys into a single public key, after that, the server will deliver proof. $Pf = \{S, pk\}$ to the auditor for audit.

$$S = \sum_{i=t_1}^{t_c} s_i(2)$$

$$P(K) = \sum_{i=t_1}^{t_c} pk_i \quad (3)$$

Response algorithm consists of the following main steps:

Algorithm Five: Response

Input: Challenge (R) , encrypted data file (E) , public key (pk) , signature (S) .

Output: Proof of possession (Pf)

- 1: If the Cloud Service Provider accepts the challenge, the following will occur:
- 2: Use Eq.(2) to produce S , and Eq.(3) to produce pk , where $t_1 \leq i \leq t_c$.
- 3: Generate $Pf = \{S, pk\}$.
- 4: Send the Proof of Possession (Pf) to the (TPA) for auditing process.

Algorithm Six_Verify Proof

During this phase, Upon receiving the proof (Pf) from the CSP, the auditor validates its authenticity by performing a verification process using Equation (4).

$$e(\sum_{i=1}^{t_c} (H(m_i)P + pk_i, S_i) = e(P, P). \quad (4)$$

If the equation (4) is true, outputs TRUE, and it returns success to the user, otherwise, outputs False, and it returns failure.

The verification algorithm consists of the following main steps:

Algorithm Six: Verify Proof

Input: Proof of possession (Pf), Challenge (R), descriptive data (D_{Detail}) .
--

Output: success, failure

1. The (TPA) conducts an audit of the data to determine its validity utilizing Equation (5) for the verification process.

2. TPA send audit result to the user.

3. The algorithm returns either a Success or Failure outcome based on the audit result.

• Enabling Dynamic Data Operations

The cloud model allows users to access and update external data regularly for different application needs [20] (Tian et al., 2019). As a consequence of this, ensuring the dynamics of the data is essential for the protection of personal information and the facilitation of public audits. In this section, this study aims to illustrate the proficiency and efficacy of our system in managing data dynamics, specifically in handling block-level updates, deletions, and insertions.

1. Modifying data blocks :

In the cloud model, modifying data blocks involves replacing specific blocks. A user requests the cloud to download the block (e_i) and generates a new private key. A new random value (P) is selected to generate a new public key (pk_i) and signature (S_i) for the updated block (e_i). The user updates the descriptive data (D_{Detail}) of the modified block with (m_i, P_i). The modified block (e_i) and values (S_i, pk_i) are uploaded to the cloud. The user requests the deletion of the old block (e_i) and its corresponding values. The updated descriptive data is also sent to the TPA.

2. Inserting Data Blocks:

Involves expanding the user-encrypted data file (E) with new user-defined blocks in the server. To insert a block after the k^{th} block, the user selects the block (e_i), generates private key (sk), and a random value (P) to create public key pk and signature S for the block. The user also creates descriptive data (D_{Detail}) for the new block. Then, the user sends block (e_i) with corresponding values (S_i, pk_i) to be added to their cloud-stored file and sends D_{Detail} to the (TPA).

3. Deleting Data Blocks:

Deleting Data Blocks involves removing a specific block from the user's cloud-stored file, causing subsequent blocks to shift forward. To delete block (e_i), the user directly requests the server to remove it, along with its associated values (S_i, pk_i). Simultaneously, a request is sent to TPA to delete block name (m_i) and its corresponding value (P_i) belonging to block (e_i).

VI. EVALUATION

Within this segment, we assess the suggested system by looking at how secure it is and how well it performs. Both the system's security and performance indicators will be reviewed as part of this assessment.

- **The security analysis**

During the suggested scheme's security review, multiple areas of security are evaluated, including system correctness, privacy preservation, secrecy, and data integrity protection.

- **Ensuring System Correctness**

In order to determine whether or not the suggested auditing system is accurate, the verification equation is checked using the theorem that is shown below.

Theorem 1: Given a data file $D = d_1, \dots, d_n$ and the encrypted data file $E = e_1, \dots, e_n$, along with signatures $S = S_1, \dots, S_n$ on E , where (n) represents the number of encrypted file blocks E , the proposed system enables the TPA to successfully demonstrate the integrity of E .

Proof: To demonstrate the correctness of the system that have been proposed, we must verify the accuracy of Equation (5). Using the bilinear pairing properties, the TPA randomly selects block names (m_i) from the encrypted file E to verify its integrity. The TPA then calculates the challenge (R) , where $R = \{(i, r_i)\}, t_1 \leq i \leq t_c$. Upon receiving the challenge R , cloud calculates the proof (Pf) , where $Pf = \{S, pk\}$, then sends it to auditor. Lastly, The auditor will verify that the proof is accurate (Pf) by proving Equation (4) as follows:

$$\begin{aligned} e\left(\sum_{i=t_1}^{t_c} (H(m_i)P + pk_i, S_i)\right) &= e(P, P) \\ &= e\left(\sum_{i=t_1}^{t_c} (H(m_i)P + pk_i, S_i)\right) \end{aligned}$$

We can get the signature $S = \frac{1}{H(m_i) + x_i} P$

, and the public key $pk_i = x * P$, therefore the following equation can be obtained

$$\begin{aligned} &= e\left(\sum_{i=t_1}^{t_c} (H(m_i)P + x * P, \frac{1}{H(m_i) + x_i} P)\right) \\ &= e\left(\sum_{i=t_1}^{t_c} (H(m_i) + x_i)P, (H(m_i) + x_i)^{-1}P\right) \\ &= e(P, P)^{(H(m_i) + x_i) \cdot (H(m_i) + x_i)^{-1}} \\ &= e(P, P) \end{aligned}$$

- **Ensuring Privacy Preservation**

The suggested solution maintains anonymity during the auditing process, preventing attackers from learning. A challenger Such as cloud server and an adversary for example, a malicious TPA are involved in the procedure. The steps are as follows:

1. The adversary receives descriptive data

$(D_{Detail} = P_i, (m_i))$ from user and selects different named blocks $(m_1, \dots, m_K)$ for the challenge.

2. The adversary creates a challenge $R = \{(i, r_i)\}$ with $t_1 \leq i \leq t_c$ for the challenged blocks and sends it to the challenger to request proof.

3. The challenger generates proof $Pf = \{S, pk\}$ for the challenged blocks and sends it to the adversary in response.

In our signature system, the hash function (H) prevents collisions. The adversary verifies block integrity by calculating $H(m_i)$ for challenged blocks names and confirms data validity with Eq. (5). The signature (S) ensures user data privacy as the adversary cannot extract any information about the data.

- **Ensuring Confidentiality**

In cloud computing, ensuring data confidentiality is crucial to prevent unauthorized access. The proposed system achieves this by using strong encryption, specifically the AES algorithm. Data is divided into blocks and encrypted with AES, providing maximum security as the key length matches the plaintext length, making it resistant to brute-force attacks. AES sensitivity to plaintext ensures even a slight key difference results in significant decryption differences, rendering incorrect keys useless. This approach ensures data confidentiality and a secure environment for cloud storage and retrieval.

- **Ensuring Data Integrity Protection**

The proposed approach for integrity protection is founded upon either the conventional Computational Diffie-Hellman (CDH) problem or the co-CDH problem. One of the potential risks that is taken into account inside this system is the threat to integrity, which arises when an unauthorised individual has access to data, signatures, and the public key. The primary goal of the assailant is to generate compelling proof pertaining to the act of data fabrication. The suggested solution guarantees the preservation of cloud data integrity by utilising the CDH assumptions. The security of the system is evaluated by means of a game that encompasses exchanges between an adversary and a challenger.

Theorem 2 states that if there is a forger F (with parameters t, qH, qs, ϵ) who can execute an adaptive chosen message attack on the proposed signature scheme, then there exists an algorithm A (with parameters t', ϵ') that can solve qs -CAA (collusion attack algorithm with qs traitors), where: $\epsilon' \geq (qs / qH)^{qs} \cdot \epsilon^1$ and $t' = t$.

Proof: In the proposed signature scheme, the signing process involves querying $H(m)$ before signing a message m . The proof is conducted in the random oracle model, where the hash function is modeled as a random oracle producing evenly dispersed outputs. Suppose there is a forger $F(t, qH, qs, \epsilon)$ that breaks the signature scheme using an adaptive chosen message attack. could use forger F to construct an algorithm A to solve qs -CAA. Algorithm A takes a challenge with parameters $P, Q = x * P, h_1, h_2, \dots, h_{qs} \in \mathbb{Z}_q$, and $(h_1 + x)^{-1}P, (h_2 + x)^{-1}P, \dots, (h_{qs} + x)^{-1}P$. The task for A is to compute $(h_2 + x)^{-1}P$ for a certain $h \notin \{h_1, \dots, h_{qs}\}$. A Assumes the role of the signer, establish P_{pub} as Q , and handles hash oracle and signing queries. F , the querying party, is assumed not to duplicate any hash or signature queries.

S_1 : A generates qH responses $\{w_1, w_2, \dots, w_{qH}\}$

for the hash oracle queries with randomly distributed $h_1, \dots, h_{qs}$ within this response set.

S_2 : F executes a hash oracle query on m_i for $1 \leq i \leq qH$, and A responds with w_i .

S_3 : F initiates a (signature oracle query) for w_i . If w_i is equal to h_j , A provides the response $(h_j + x)^{-1}P$. Otherwise, the process terminates, and A is unsuccessful.

S_4 : A uses the response $(h_j + x)^{-1}P$. Finally, F terminates and produces a valid message-signature pair (m_i, S_i) where $H(m) = w_1$, fulfilling the equation: $e(H(m)P + Q, S) = e(P, P)$.

- **The Performance Analysis**

The evaluation of the performance for the proposed system is crucial in order to determine its efficiency from various perspectives, including user experience, auditor evaluation, and the viewpoint of the cloud service provider. This study aims to analyze the impact of communication and computation overhead on the optimal performance of a system.

1. Computational overhead : The expenses associated with generating keys and signatures, as well as downloading and uploading blocks from the cloud, comprise a portion of the overall cost of computing borne by the user. Additional costs will be incurred for the production of the evidence by the CSP as well as the cost of having the proof verified by the third party auditor.

2. Overhead costs related to communication: These charges encompass expenses related to data volume and bandwidth requirements essential for communication between the Cloud Service Providers (CSP) and the Third Party Auditor (TPA).

Python was used as the programming language to implement the proposed system on a Windows 7 operating system: Intel (R) Core (TM) i7-8550U CPU @ 1.80GHz, 1.99 GHz, and 8GB memory. The system's functionality was tested using transactions from the Berka Banking Dataset [21]. The dataset used in testing represents the original file, divided into blocks ranging from 100 to 500 blocks, each with a size of 1 kilobyte. Testing was conducted on the AWS cloud system for data deployment and performance evaluation.

- **The computational overhead**

Burden of the proposed system is examined and assessed from three distinct perspectives: the user's standpoint, the Third-Party Auditor's (TPA) standpoint, and the Cloud Service Provider's (CSP) standpoint. The comprehensive nature of this method ensures that the recommended system will undergo thorough evaluation across all domains, utilizing the proposed algorithms to achieve optimal levels of safety and efficacy. On the user's side, the data is depicted in Figure 2 as being segmented into a variety of data blocks, each of which has a unique number. As shown in figure 2, the system under consideration is able to partition 100 data blocks in a shorter amount of time than 0.0040 seconds. The proposed system employs AES encryption on the user's side to ensure data confidentiality after dividing it into multiple block .

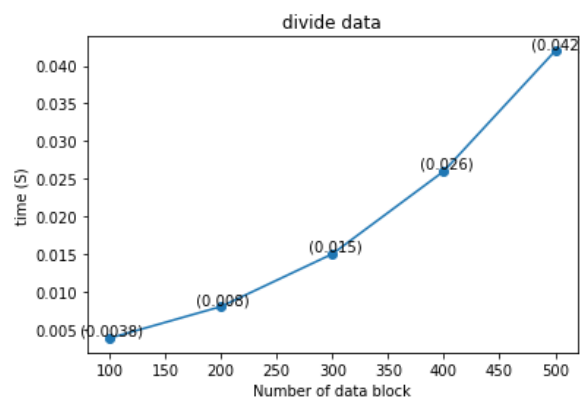


Fig. 2. The computation overhead of divide data

Figure 3 shows a comparison of computational costs between the system that proposed and Shinde et al. (2019)[22]. Encrypting 500 KB of data takes less than 0.16 seconds with our system, Although Shinde et al. (2019)[22] requires 0.57 seconds for the same task. The proposed system's cryptographic expense remains appealing as it enhances data security without burdening the user's resources significantly. User-generated key generation for 100 blocks takes 0.0014 seconds (Figure 4).

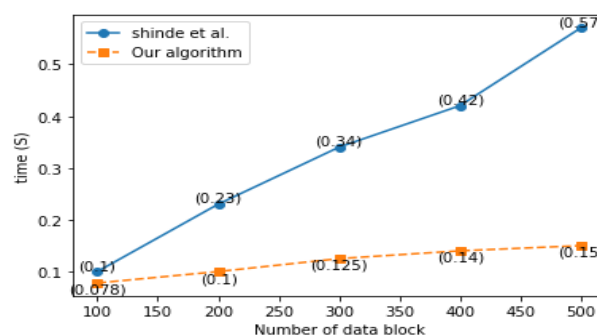


Fig.3. The computation overhead of encryption

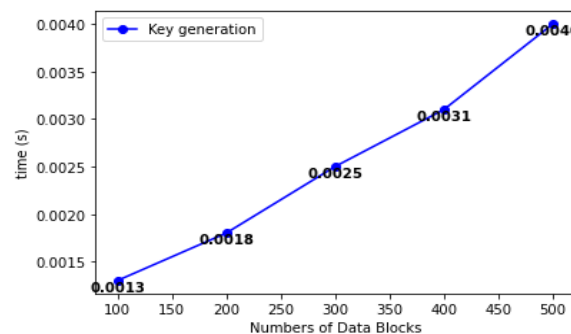


Fig.4. key generation

Figure 5 highlights that our proposed system has significantly lower computational overhead for signature generation compared to RSA [4], BLS [2][3] and ZSS [23]. For 500-block data files, our scheme takes 0.0055 seconds, whereas RSA, BLS and ZSS require 16.22 second, 10.1621 seconds and 3.0699 seconds, respectively.

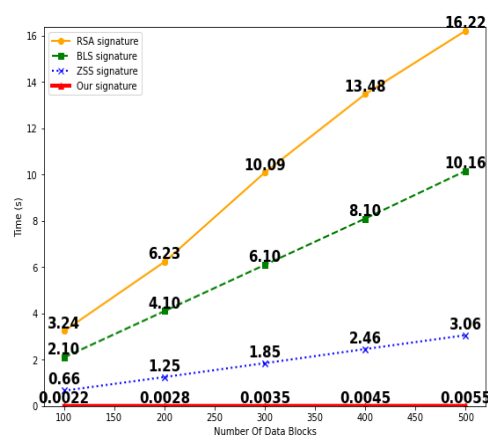


Fig.5. Signature generation time

In addition to the audit process, the auditor generates challenges, while the proof generation process occurs solely on the server-side. It is noteworthy that challenge generation is fast, requiring only 0.0002 seconds for every 100 KB of data. Conversely, the response and proof verification processes take longer. In order to enhance efficiency in terms of time allocation, it was determined that the quantity of challenge blocks would be augmented from 100 to 500. The results depicted in Figures 6 and 7 demonstrate that as the number of challenges increases, the time required to generate challenges, respond to them, and verify the proofs also increases.

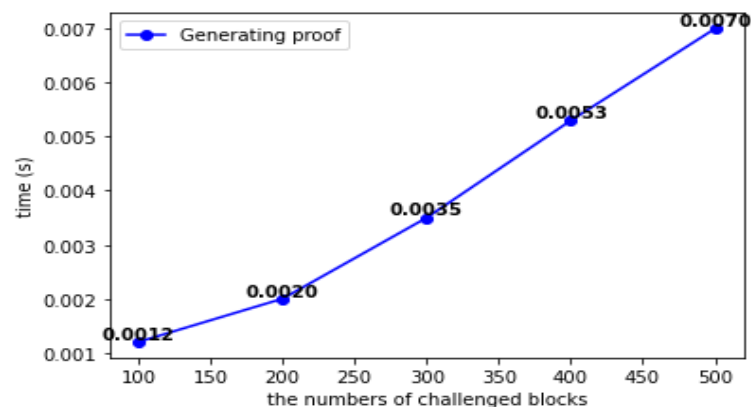


Fig.6 . The time Required for Generating Proof

This experimental analysis is appropriate because as the number of challenged blocks rises, generating the necessary random values for challenges becomes more complex.

Additionally, the cloud infrastructure experiences increased computational load while the auditor handles additional processes Figure 8 compares the computation overhead in a scenario where the TPA randomly selects 200 blocks to verify integrity of user's outsourced data. The results clearly indicate that the evaluated scheme has lower computation costs during the Challenge and Verify Proof phases compared to the RSA scheme [4]. The user obtains the original data file by downloading chunks from the cloud and decrypting them. Figure 9 depicts the situation, the data decryption in our proposed system takes about 0.18 seconds for 500 KB of data. In contrast, Shinde et al. (2019)[22] needed 1.09 seconds for the same data decryption, showing our system's lower computational cost. Fig. 10 elucidate the procedure involved in amalgamating decoded blocks in order to restore the integrity of the initial data file.

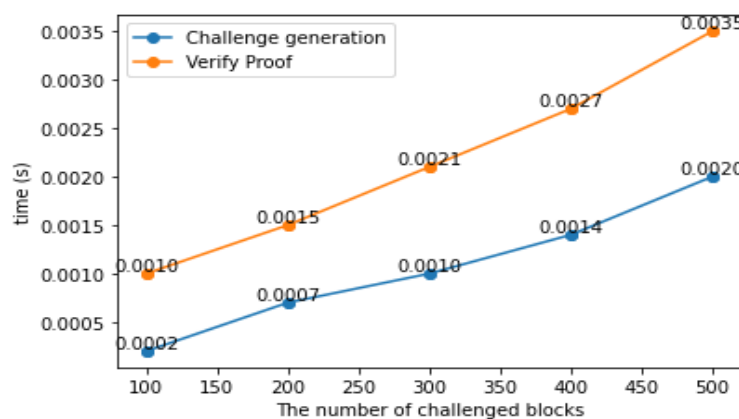


Fig.7. The time needed to create a challenge and examine the evidence

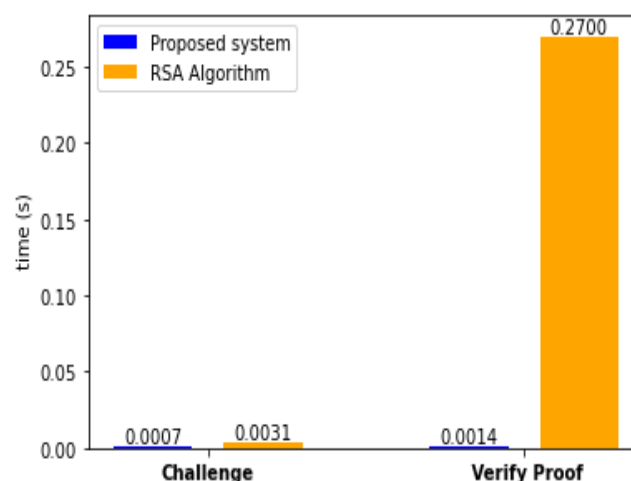


Fig.8. Computational Overhead Comparison

Regarding data dynamics, we can determine the computational expenses associated with inserting a new block. When a user wishes to add a new block (1 kilobyte) to their dataset within cloud computing, they employ the encryption algorithm AES to encrypt this block. Subsequently, The encrypted block must be put in the user's data file at a certain position. To facilitate this process, the user generates a private key (sk) and a public key (pk), followed by creating a signature (S) for the encrypted block. Additionally, descriptive data ($\mathcal{D}_{Detail}$) is created for this block. As a result, the cost of computing to encrypt the selected block is (0.0032), while the

costs of computing to generate keys and signatures are (0.00015) and (0.00020), respectively. If the user wants to modify a block, they start by asking for the block to be downloaded from the cloud. The user then goes on to encrypt the altered block, produce the private and public keys, and produce a signature for the block after it has been downloaded. Additionally, before sending the updated block back to the cloud, the user will update any descriptive data that may have been affected by the change. During the process of inserting blocks into the cloud, it is made abundantly clear that the timestamps for the generation of encryption keys, generation of signatures, and generation of encryption keys all remain unchanged.

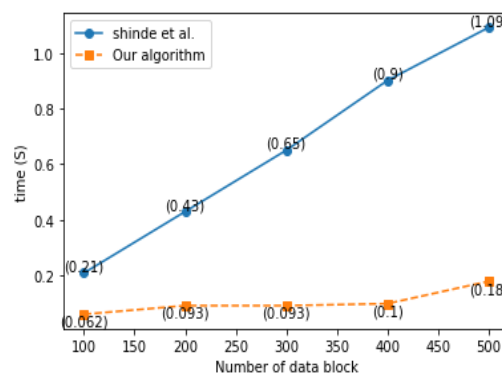


Fig. 9. The cost of decryption computation

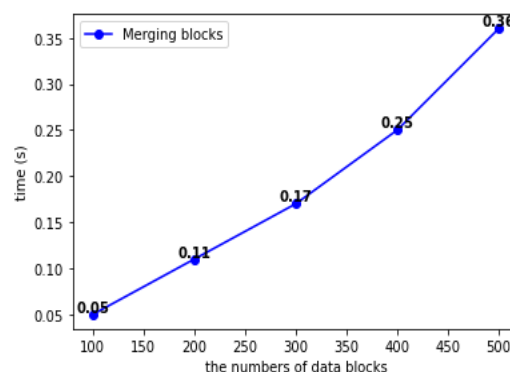


Fig.10. Block merging calculation overhead

However, it is possible to calculate the time taken to download the block, make modifications, and re-upload it to the cloud, which amounted to 2.145 seconds. In the case of removing a particular block, the user makes a cloud request. It takes the cloud service provider (CSP) 0.81 seconds to search for the block that has to be erased, and 0.20 seconds to actually delete the block. As a result, this deletion operation took 1.017 seconds in total, including the search and deletion.

• Communication Overhead

Communication overhead in the proposed system is due to data transmission to and from the cloud, as well as the challenge message " R " sent to the CSP and its response for auditing. The system chooses random blocks for testing, and double submissions are used in the audit process. The first submission includes the challenge blocks represented as $R = \{i, r_i\}$, and the number of blocks sent for auditing is denoted as 'n', resulting in a communication cost of $O(n)$. The second submission is the TPA's response, represented as proof $Pf = \{S, pk\}$. To evaluate the system, experiments were conducted to measure duration necessary for the process of uploading and downloading a single file or many files from a cloud storage system. The data file was divided into multiple

blocks and individually encrypted, and the results for upload and download times are depicted in Figures 11 and 12, respectively. The average communication time remains steady for smaller block sizes, but as the number of blocks increases, the workload intensifies. Blocks are typically faster to upload than download. In conclusion, The number of blocks increases in direct proportion to the time required to upload them to the cloud and download them from there. In the case of dynamic data, it takes one second to upload a new block into the cloud that is one kilobyte in size and one second to retrieve the block out of the cloud.

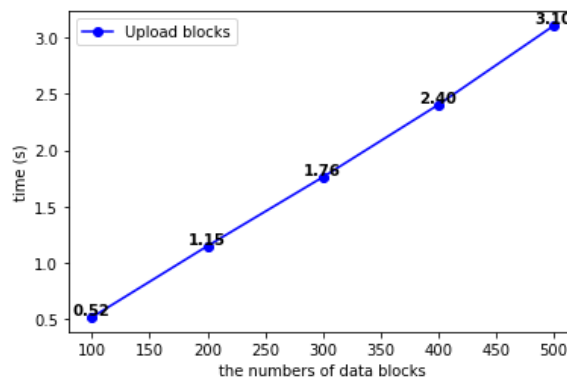


Fig. 11. The upload blocks

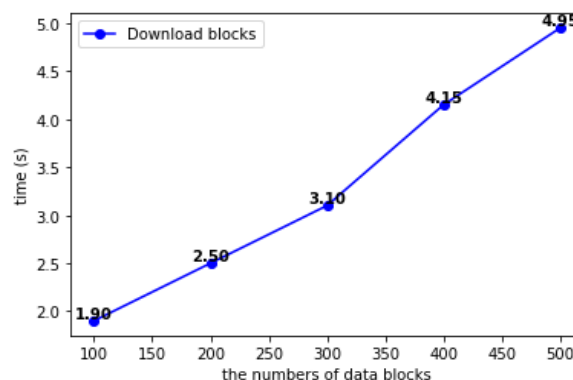


Fig. 12. The download blocks

VII. CONCLUSION

In the current research, an efficient scheme for verifying the integrity of cloud data was developed, which was based on a new short signature mechanism. In addition to ensuring data integrity, the proposed technology protects data privacy by encrypting it before it is sent to the cloud. In this study, a data integrity verification system for the cloud is presented that, in comparison to the BLS and RSA techniques, delivers a greater level of efficiency. By utilizing the new short signature algorithm, the computational overhead associated with the hash function during the signature process is reduced compared to the BLS algorithm. The results of experiments show that this method has a lower computational and communication overhead compared to existing data integrity solutions based on RSA and BLS. Moreover, our scheme incorporates support for public auditing by introducing a TPA. provide evidence that the method is secure against adaptive selection message assaults in line with the random oracle model, as required by the CDH difficulty assumption. It also provides support for dynamic data operations, including addition, deletion, and modification.

REFERENCES

- [1] M. Suganya and T. Prabha, "A Comprehensive Analysis of Data Breaches and Data Security Challenges in Cloud Environment," *Available SSRN 4111762*, 2022.
- [2] R. Li, H. Yang, X. A. Wang, Z. Yi, and K. Niu, "Improved Public Auditing System of Cloud Storage Based on BLS Signature," *Secur. Commun. Networks*, vol. 2022, 2022.
- [3] X. Luo, Z. Zhou, L. Zhong, J. Mao, and C. Chen, "An effective integrity verification scheme of cloud data based on BLS signature," *Secur. Commun. Networks*, vol. 2018, pp. 1–11, 2018.
- [4] Z. Xu, L. Wu, M. K. Khan, K. K. R. Choo, and D. He, "A secure and efficient public auditing scheme using RSA algorithm for cloud storage," *J. Supercomput.*, vol. 73, no. 12, pp. 5285–5309, 2017, doi: 10.1007/s11227-017-2085-8.
- [5] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," in *Public Key Cryptography—PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. Proceedings 7*, Springer, 2004, pp. 277–290.
- [6] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22*, Springer, 2002, pp. 354–369.
- [7] A. Juels and B. S. Kaliski Jr, "PORs: Proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 584–597.
- [8] G. Ateniese *et al.*, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 1–34, 2011.
- [9] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," *Cryptol. ePrint Arch.*, 2008.
- [10] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1408–1421, 2019.
- [11] L. Chen, "Using algebraic signatures to check data possession in cloud storage," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1709–1715, 2013.
- [12] Y. Sun, Q. Liu, X. Chen, and X. Du, "An adaptive authenticated data structure with privacy-preserving for big data stream in cloud," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3295–3310, 2020.
- [13] Y. Ping, Y. Zhan, K. Lu, and B. Wang, "Public data integrity verification scheme for secure cloud storage," *Information*, vol. 11, no. 9, p. 409, 2020.
- [14] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues, "Cloud centric authentication for wearable healthcare monitoring system," *IEEE Trans. Dependable Secur. Comput.*, vol. 17, no. 5, pp. 942–956, 2018.
- [15] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Computer Security—ESORICS 2009: 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings 14*, Springer, 2009, pp. 355–370.
- [16] N. Garg, S. Bawa, and N. Kumar, "An efficient data integrity auditing protocol for cloud computing," *Futur. Gener. Comput. Syst.*, vol. 109, pp. 306–316, 2020.
- [17] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *2010 proceedings ieee infocom*, Ieee, 2010, pp. 1–9.
- [18] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, pp. 297–319, 2004.
- [19] Y. Li, Y. Yu, B. Yang, G. Min, and H. Wu, "Privacy preserving cloud data auditing with efficient key update," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 789–798, 2018.
- [20] H. Tian, F. Nan, H. Jiang, C.-C. Chang, J. Ning, and Y. Huang, "Public auditing for shared cloud data with efficient and secure group management," *Inf. Sci. (Ny)*, vol. 472, pp. 107–125, 2019.
- [21] A. Phil, D. Marguerite, and K. Priya, "Credit Card Analysis of Czech Bank." 2002.
- [22] M. Shinde, A. Jaiswal, A. Shinde, and M. R. R. Nikam, "Storage Security in Cloud Computing," 2019.
- [23] H. Zhu *et al.*, "A secure and efficient data integrity verification scheme for cloud-IoT based on short signature," *IEEE Access*, vol. 7, pp. 90036–90044, 2019.