Research Article

Akshay KC, Balachandra Muniyal*, and Vikalp Parashar

Optimizing data retrieval for enhanced data integrity verification in cloud environments

https://doi.org/10.1515/eng-2024-0058
received November 27, 2023; accepted June 07, 2024

Abstract: In today's rapidly evolving digital landscape, the urgency to secure data within expansive cloud storage systems has reached unprecedented levels. Conventional remote storage methods, while widely used, are inherently vulnerable to security breaches, corruption, and tampering. Recognizing this critical challenge, a state-of-the-art protocol has emerged to address these vulnerabilities head-on. This innovative solution integrates a sophisticated binary search tree (BST) structure with elliptic curve cryptography, ensuring not only efficient data retrieval but also robust encryption mechanisms. The protocol goes further by meticulously computing secure hashing algorithm hash values to verify the integrity of files, leaving no room for unauthorized modifications or tampering attempts. A thorough comprehensive benchmarking analysis has been conducted comparing this protocol with established techniques such as Rivest, Shamir, Adleman encryption and doubly linked list-based index table structures. The findings reveal that the proposed protocol outperforms these conventional methods, showcasing superior security features and computational efficiency. Remarkably, the proposed method reduces overheads by an impressive 5%, making it a highly favorable choice for both businesses and academic institutions. This marks a significant advancement toward fortified data security in cloud environments, contributing substantially to the ongoing discourse on secure data storage and management.

Keywords: cloud security, information security, cloud auditing, cloud computing, cryptography, elliptic curve encryption, data integrity verification

*** Corresponding author: Balachandra Muniyal,** Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal – 576104, Karnataka, India, e-mail: bala.chandra@manipal.edu
Akshay KC: Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal – 576104, Karnataka, India,
e-mail: akshay.kc@manipal.edu
Vikalp Parashar: Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal – 576104, Karnataka, India,
e-mail: vparashar0299@gmail.com

1 Introduction

The escalating growth of data transfers due to increasing internet connectivity has led to an unprecedented surge in data production and exchange. Over the past 5 years (2020–2024), the volume of generated data has exceeded that of the entire preceding decade (2010–2019), with projections indicating a further exponential increase. By 2025, the global data creation is anticipated to surpass 181 zettabytes, as depicted in Figure 1.

This rapid expansion in data creation has posed significant challenges for organizations in meeting their data storage requirements. As a result, cloud storage has emerged as a viable solution, albeit accompanied by new security considerations that demand careful attention. With a substantial adoption of cloud computing among enterprises, the importance of robust cloud security measures cannot be overstated. Forecasts by Gartner suggest a notable growth of 23.1% in the global market for public cloud services by 2025 [1].

Despite the advantages offered by cloud-based environments, concerns persist among information technology professionals regarding the secure transfer of sensitive data to the cloud, encompassing issues related to security protocols, governance, and regulatory compliance. The potential risks of inadvertent data exposure or cyber attacks leading to the compromise of confidential corporate data and intellectual property are significant and require proactive mitigation strategies.

In this context, data auditing plays a pivotal role in evaluating the accuracy, reliability, and security of data throughout its lifecycle. It helps to prevent the spread of poor data quality and security issues throughout an organization [2]. There have been numerous methods proposed for dynamic auditing in the past, including those proposed in previous studies [3–7]. While these methods claim to reduce computational cost, the communication cost remains

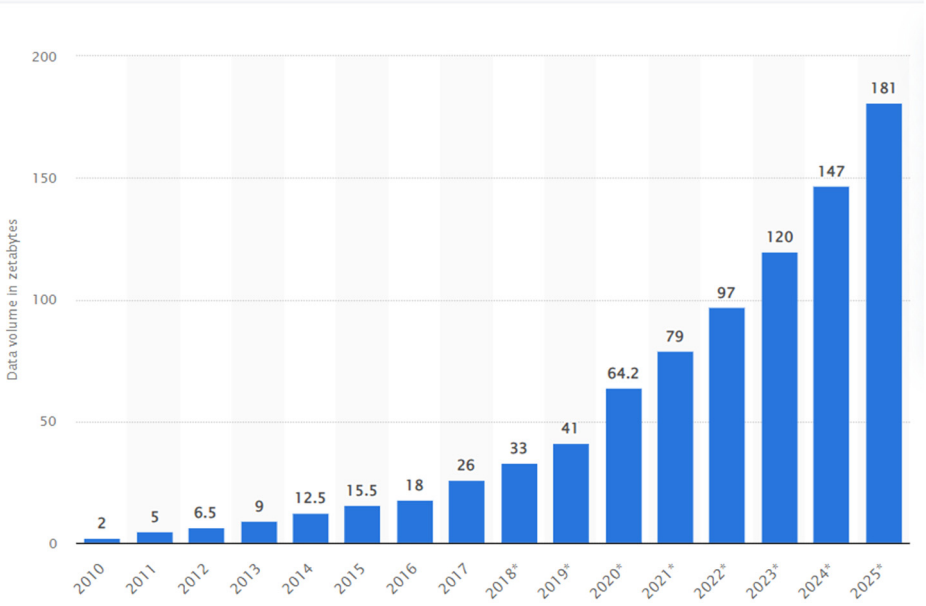

Figure 1: Exponential growth in the amount of data created by 2025 [8].

a challenge in these methods. Hence, there is a pressing need to devise an enhanced model for data integrity audits that effectively balances data security and accessibility. The proposed binary search tree (BST)-based model aims to achieve this equilibrium by implementing a robust encryption mechanism to ensure the security and confidentiality of information stored on remote servers. This approach facilitates secure data storage while facilitating essential data integrity audits, thereby addressing critical concerns and advancing best practices in cloud-based data management and security.

The ultimate goal is to provide a solution that enables organizations to securely store their data on remote servers with confidence, knowing that their information is safeguarded by robust encryption mechanisms and that regular data audits can be conducted to ensure data integrity. In the context of escalating reliance on cloud-based storage solutions, effective data security measures have become increasingly critical. The proposed BST-based model directly addresses this need, offering a practical and efficient solution for organizations looking to protect their sensitive information in the cloud. By optimizing data retrieval processes and enhancing data integrity verification mechanisms, it will contribute significantly to the field of cloud data management and security. It underscores the relevance and importance of the research in optimizing data retrieval processes to achieve enhanced data integrity verification in cloud environments, thereby addressing key challenges and making substantial contributions to the field.

1.1 Research contribution

The proposed study presents a comprehensive contribution to the field of data integrity verification in cloud environments. The key contributions are highlighted as follows:

1) A novel system for verifying the integrity of data stored in the cloud environment is devised. This system is a significant improvement over the existing methods and enhances the overall reliability of cloud storage systems.
2) In the proposed approach, the BST data structure for data insertion, deletion, updation, and auditing has been utilized in the cloud environment. This leads to improved performance and efficiency in managing the data stored in the cloud.
3) The work described in this article has been thoroughly analyzed for various security threats, and countermeasures have been proposed to mitigate these risks. This ensures that the data stored in the cloud is protected against unauthorized access, tampering, and other security threats.
4) To confirm the effectiveness of the proposed method, a statistical analysis is conducted, which shows that the proposed approach outperforms the existing works in terms of data integrity verification in the cloud environment.

1.2 Organization of this article

Section 2 reviews relevant literature on data auditing protocols, summarizing prior research efforts aimed at enhancing data storage and protection on remote servers. In

Section 3, the key research contributions are highlighted showcasing novel approaches, innovative techniques, and insights addressing existing literature gaps to provide new perspectives for future work. Section 4 provides essential background knowledge for implementing the proposed approach, explaining preliminary concepts and secure assumptions to establish a strong foundation for understanding later technical details. Section 5 thoroughly examines the proposed method, breaking it down into parts covering system design aspects, workflow, architecture, and visual aids to elucidate component interactions. It offers a comprehensive understanding of how the method addresses challenges and requirements outlined in Section 1. Section 6 provides the assessment of security threats to the proposed system. It also offers a detailed analysis of mitigation measures emphasizing the importance of the robust security for data protection and system integrity. Section 7 compares the components of the proposed model with other existing models, highlighting the advantages and superiority. Section 8 offers a comprehensive conclusion, summarizing findings, strengths, and future potential of the proposed method. It also identifies areas for improvement and extension.

2 Literature survey

This section details about the past works and recent advancements made in the domain of data integrity check in the cloud environment. These works have been categorized under various schemes or methods used for integrity verification process. The goal is to present contemporary literature in the field, discuss its design and implementation, and highlight the research gaps that have necessitated this work.

2.1 Conventional schemes

Ateniese *et al.* [9] introduced the concept of the "provable data protocol (PDP)," which allows a client to verify that the original data are stored on an untrusted server without retrieving it. On the other hand, Juels *et al.* [10] proposed a work called "proof of retrievability (PoR)," which not only verifies the integrity of the data but also ensures that it can be retrieved from the cloud through an error-correcting method.

Liu and Zic [11] introduced a PoR scheme built on homomorphic encryption. The PoR scheme allows for the retrieval of homomorphically encrypted data through the generation of probabilistic and homomorphic message authenticators, allowing for direct handling of the encrypted data by the cloud. The integrity of outsourced computations can also be verified by the PoR scheme. The researchers developed a prototype to test the scheme; however, no comparison was made to any existing work.

2.2 Schemes based on Merkle hash tree (MHT)

Wang *et al.* [4] proposed a protocol that balances both public auditing and dynamic information management. It uses a combination of Boneh–Lynn–Shacham (BLS) [12] signatures and MHT. While this protocol ensures data integrity, it falls short in providing privacy for information stored in remote storage.

Luo [13] improved the prior privacy-preserving model and proposed an effective integrity check technique of cloud data based on signature (BLS) that secures both public access and data privacy.

Du *et al.* [14] proposed a cost-effective solution for file storage that combines proof of ownership and retrievability. The files are encoded using erasure coding, and the use of Merkle trees and homomorphic verifiable tags reduces computation costs, particularly for large files. However, this approach increases storage costs.

2.3 Schemes with third-party auditor (TPA)

Kwon *et al.* [15] presented a review system for dynamic distributed information in cloud storage, which involves a third-party reviewer and a record table. The uploader must split the information into chunks, label them, and direct them to the clients for sharing. The uploader can also confirm the information with the third-party inspector. However, this approach incurs high communication costs due to the frequent exchange of messages between clients and data uploaders, and this also results in increased storage expenses for the third-party inspector. Every time the information is updated, the record table must also be updated by the third-party inspector.

More and Chaudhari [16] presented a strategy in which information is encrypted at the owner's side by dividing it into chunks and applying scrambling, producing hash labels for each chunk. The hash labels are then used to generate a signature for the specific owner. However, this approach requires more time and involves multiple

steps on the owner's side. The proposed scheme also involves verification by a third-party analyst, who must recreate the steps performed by the owner to cross-verify the digital signature, adding an additional layer of security.

The aim of Shah *et al.* [17] was to develop and introduce a TPA in cloud storage to enhance security and maintain data integrity. The TPA is noted for its adaptability and ease of use. The authors claimed that the TPA can effectively perform simultaneous reviews for multiple clients and verify the integrity of information stored in the cloud. The study suggests that the proposed protocol is secure and well structured; however, the computational cost needed for implementation is not explicitly stated in this article.

2.4 Schemes without TPA

Yuan *et al.* in their study [18] proposed a strategy to enhance integrity of information sharing activities in the cloud. This strategy takes into account various important aspects such as client disavowal, open examination, multi-user adjustment, increased error detection probability, and efficient computational and communication auditing execution. The proposed approach is designed to resist client impersonation attacks and enables multiple clients to collaborate on data without the risk of impersonation or data theft or modification. However, the computational cost of implementing the proposed protocol is not specified in this article.

Zargad *et al.* [19] presented a dynamic encryption approach for cloud storage without the use of a neutral reviewer. According to the proposed method, the judgment of the information is lost when it is outsourced to the cloud. Whenever there is a security threat, an encryption method is applied based on the type and size of the information stored. The method employs an executor, such as a proxy server, to handle incorrect additions or alterations, but it lacks a proper mechanism for key management, which is an important aspect of secure information storage in the cloud.

Kaaniche in her dissertation [20] proposed a design for ensuring the integrity of information in a public network sharing setup. She proposed a system called "cloudasec" that verifies information ownership using set homomorphic verification and zero-knowledge proof techniques. Although the proposal highlights the importance of remote information checks in a cloud environment, it lacks a comprehensive overview of dynamic information corroborations.

Li *et al.* [21] presented a secure and lightweight information sharing scheme for mobile computing. They used ciphertext policy-attribute-based encryption (CP-ABE) in a cloud environment as an access control technique. The implementation was based on the access control tree concept, and the team used the description fields to implement the lazy revocation method. The study reported that the overhead or processing time is reduced on mobile devices when information is distributed in the mobile cloud environment. However, the computational cost required to implement the protocol is not clearly stated.

Anisetti *et al.* in their work [22] aimed to provide a persistent cloud service certification framework that would meet non-functional requirements as defined by a certification model and specialist. The framework relies on a chain of trust, established through the comparison of the requirements and models. However, the authors do not address the computational and communication cost required to implement the certification scheme.

Mrinal *et al.* [23] proposed a technique to maintain the authenticity of information during transmission by hiding the information behind an image. The approach focuses on ensuring the integrity of the information, but it does not provide a solution for the confidentiality of the information.

Sun *et al.* [24] presented an open data integrity verification scheme that uses cryptographic indistinguishability obfuscation. The scheme aims to hide users' secret data from the cloud server by embedding it into a scrambled program. However, the computational cost required to implement this scheme is not mentioned in the study.

Wang and Di [25] presented a multi-agent-based cloud storage system that implements a multi-copy data integrity check method. They utilized the bi-linear mapping method to generate the keys, and a multi-branch tree for authentication was proposed to perform signing, validation, and confirmation through multi-copy data signatures. The allocation of resources and workflow was based on quality of service (QoS) and represented through a directed acyclic graph (DAG). The jobs were scheduled based on the QoS preference settings.

Ganesh and Manikandan [26] presented a scheme for verifying the remote information stored in public clouds, specifically designed for mobile users. They claimed that the communication and computation overhead was reduced compared to previous works. The authentication process was carried out during the auditing, modification, deletion, and insertion of blocks.

Chidambaram *et al.* [27] proposed a scheme for secure storage of customer data in the cloud. They utilized the Rivest, Shamir, Adleman (RSA) algorithm and digital fingerprint generated using the MD5 hash function. The authors claimed that data encrypted with the RSA algorithm cannot be modified by a third party.

Zhong *et al.* [28] developed an information-theoretic safe proof of ownership mechanism for file rating. The K-means technique was integrated with file rating, and the use of random seed technology and pre-calculation approach was proposed to provide a secure and fast proof of ownership mechanism.

2.5 Schemes with other structures

Canto *et al.* [29] presented the methods for creating secure key generators for code-based post-quantum cryptosystems on field programmable gate arrays (FPGA) platforms. It focuses on ensuring reliable key generation, critical for system security. The study discusses optimizations leveraging FPGA parallelism. However, it may not extensively cover potential limitations, such as resource constraints or performance trade-offs, inherent in FPGA-based implementations.

Chen *et al.* [30] put forth a groundbreaking public auditing protocol based on the adjacency-hash table. This new protocol was designed to be more efficient in terms of dynamic auditing and data updating when compared to the existing methods available at that time. The authors claimed that this protocol outperformed the state of the art methods in these aspects.

Koziel *et al.* [31] discussed a highly efficient implementation of the supersingular isogeny Diffie–Hellman key exchange protocol on advanced RISC machine (ARM) processors. It focuses on optimizing the protocol's performance on ARM-based devices, which are commonly used in mobile and embedded systems. The study likely covers techniques and optimizations to leverage ARM's neon technology for enhanced performance. However, a potential limitation of this work could be its applicability to specific ARM architectures or limitations in scalability to larger systems beyond mobile or embedded devices.

Kermani *et al.* [32] explored integrating security research and education for new medical devices. It addresses interdisciplinary strategies for tackling security concerns. However, a limitation could be the complexity of merging diverse expertise, such as medical, engineering, and cybersecurity, which might hinder comprehensive security solutions, and on the other hand, Kermani [33] focused on detecting faults in very large scale integration implementations of advanced encryption standard (AES) encryption. It presents schemes to mitigate faults affecting AES' performance, but a limitation may arise from the trade-offs between fault detection accuracy and overhead. More robust detection methods could demand additional hardware resources, impacting efficiency in terms of performance or area.

Yan [34] proposed a secure and efficient data exchange solution for dynamic user groups. To ensure user identification tracking and handle the addition and removal of dynamic users, Yan introduced the concept of a rights distribution center (RDC). The RDC role was designed to protect the privacy of user identities during a third-party audit of data integrity, thereby increasing the fairness of the audit and establishing a new approach to verifying the integrity of shared cloud data.

Chandel *et al.* [35] conducted a study comparing the time complexity of RSA and elliptic curve cryptography (ECC) algorithms for encrypting and decrypting data. They found that the time required for RSA grows logarithmically as the amount of data to be encrypted increases, while the time complexity of ECC remains relatively constant.

Niasar *et al.* [36] focused on enhancing the performance of ECC specifically for Curve448. It discusses optimizations in hardware architectures or algorithms to speed up ECC operations on Curve448, which is crucial for secure communications protocols. However, previous studies [37–40] explored dependable architectures for finite field multipliers, emphasizing their implementation using cyclic codes on FPGA platforms. These architectures are vital in both classic and post-quantum cryptography systems. All of these articles contribute to improving the efficiency, reliability, and security of cryptographic operations, addressing key challenges in modern cryptographic implementations.

Mozaffari-Kermani *et al.* [41] discussed crucial aspects such as protecting sensitive medical data, securing medical devices, and fortifying healthcare infrastructure against cyber threats. The collection aims to underscore the growing significance of security in biomedical contexts and offers insights into current challenges and potential solutions within this dynamic field, whereas Karam *et al.* [42] discussed the significance of hands-on learning in hardware security, especially during a time when traditional in-person education is limited. However, a limitation of the study is that it may not yet have comprehensive results or evaluations due to its work-in-progress nature. Nonetheless, it offers valuable insights into innovative strategies for addressing educational challenges in hardware security during the pandemic.

The field of cloud data integrity has seen a significant amount of research over the years, resulting in the proposal of various methods and works aimed at ensuring the accuracy and reliability of data stored in the cloud. However, many of these works only focused on verifying the integrity of static data, while others relied on TPAs to carry out the check. This means that the owner or user of the data must first request the integrity check, which then triggers the TPA to conduct the audit. This process also

involves a significant amount of communication between the TPA and the owner to verify the ownership of the data and retrieve it, adding to the overall cost of the data integrity check. The BST-based model presented in this article aims to enhance the integrity check process while simultaneously reducing communication costs.

3 Background

Users may have significant assets stored in the cloud. Cloud service providers (CSPs) offer ample storage space and large-scale computing mechanisms to support these assets. However, to ensure the integrity and reliability of the data, cloud auditors are tasked with auditing on behalf of the users and providing fair and sincere results. Dos outsource their data to CSPs to take advantage of reliable remote storage and high-performance services while reducing their own storage and maintenance overhead. Since the data are stored in the cloud rather than the owner's local systems, the owner must rely on the CSP to maintain the integrity and properness of the data.

Although cloud auditors are considered trusted and credible parties, there is always the possibility of them being curious about the private information of users or owners, despite their fair and credible audit practices. This curiosity may lead them to attempt to deduce the contents of the data being stored.

Additionally, CSPs are not always trustworthy. They may have various reasons, such as protecting their reputation or gaining benefits, to not reveal the loss or manipulation of data. In some cases, CSPs may even attempt to launch attacks on cloud auditors.

However, for the work discussed in this article, there are some secure assumptions made that serve as the foundation for the security of the proposed work.

- *Computational Diffie–Hellman (CDH) assumption:* Let G be a multiplicative cyclic group of a large prime order p, where g is a generator of G and $x, y \in Z_p$. It is considered computationally intractable to compute g^{xy} when g^x and g^y are given. Specifically, for any probabilistic polynomial-time adversary ϕ, the probability of solving the CDH problem is negligible, as expressed by the inequality:

$$P(\phi_{\text{CDH}}(g, g^x, g^y \in G) \Rightarrow g^{xy} \in G : \forall x, y \in Z_p) \leq \varepsilon. \quad (1)$$

  This indicates that the security of the system is based on the assumption that CDH is a difficult problem to solve.
- *Discrete logarithm (DL) assumption:* assuming G is the multiplicative cyclic group of a large prime order p, where g is a generator of G, and given a value k such that $k = g^x$, where $x \in Z_p$, it is considered computationally intractable to determine x. In other words, the probability of solving the DL problem is negligible for any probabilistic polynomial-time adversary ϕ, as expressed by the inequality:

$$P(\phi_{DL}(g, k \in G) \Rightarrow x \in Z, k = g^x) \leq \varepsilon. \quad (2)$$

  This indicates that the security of the system is based on the assumption that solving the DL problem is difficult.

Moreover, the proposed work employs elliptic curve-integrated encryption scheme as the encryption technique to safeguard the files. A random generator point G constrained by n (the order) is selected such that G has a double, *i.e.*, if we add G to itself, the new point created must lie on the curve; G must also be additive, such that if another point F on the curve is added to G, the resultant point must likewise be on the curve. Finally, G should have an inverse, *i.e.*, given the symmetry of the elliptic curve about the x-axis, if G lies on the positive y-axis, it should be projected to the negative y-axis on the curve and *vice-versa*. Now, a huge integer K, where K is the system's private key, is used as the multiplicand and multiplied by G.

K multiplied by G is equivalent to G being added to itself K times, and the resultant point P is the public key (PK) used to encrypt the message. The elliptic curve taken in this model is secp256r1. The following is the general function of the elliptic curve:

$$y^2 = x^3 + ax + b.$$

And, this here is the secp256r1 curve $y^2 = x^3 + 7$, where $a = 0$ and $b = 7$

$$n = 115792089210356248762697446949407573529996955224135760342422259061068512044369$$

$$G = \{x = 48439561293906451759052585252797914202762949526041747995844080717082404635286,$$
$$y = 36134250956749795798585127919587881956611106672985015071877198253568414405109\}.$$

Even though there are various cryptographic hash functions designed to generate fixed-size hash values (digests) from input data such as BLAKE [43] in quantum cryptographical aspects, the proposed work uses secure hashing algorithm (SHA)-2. It is a standardized method and has a higher level of compatibility due to its widespread adoption, and integration into protocols and standards.

4 Methodology

This section describes the methodology that is followed to realize the proposed work considering the background mentioned in Section 3.

4.1 Architecture of BST-based model

In the proposed BST-based model there are three primary entities: DO, CSP, and auditor. Figure 2 depicts the overall architecture of the BST-based model with the functions of each primary entity.

- **CSP:** The CSP is a remote storage unit whose only responsibility is to store the received files. It provides the response when it is challenged by the auditor in reference to a file. When the file is requested by the auditor, it sends the corresponding file to the auditor. The BST model begins with CSP launching its storage server, as shown in Figure 3, and goes to listening mode.
- **DO:** This module represents the user side of the model. It manages all the invocation of input-output operations for the user, including insertion, deletion, update, and audit. Additionally, the DO will encrypt a new file using its private key generated using elliptic curve encryption technique before calculating the SHA-2 hash.
- **Auditor:** This module is the system that performs the auditing process. It acts as a mediator and interaction point between the DO and CSP. It also does most of the work in the BST model. It manages the insertion, deletion, and modification of files stored in CSP. It records the user's user ID (UID), IP address, and password for authentication purposes. It further maintains the file metadata for each user, including hash, version number, and file ID (FID). The storage of these data is performed using BST of BST method, which is explained in Section 4.2. This module initializes the BST by pulling data from the database and starts the server to go to the listening mode as shown in Figure 4.
- If a user submits an audit request, the auditor sends a challenge to the CSP, which in turn sends the response of the existence of the file and then sends the requested file to the auditor. The auditor in turn calculates the SHA-2 hash and compares it with the hash stored in the system. Based on the result of this comparison, the auditor informs the user whether the file is safe.

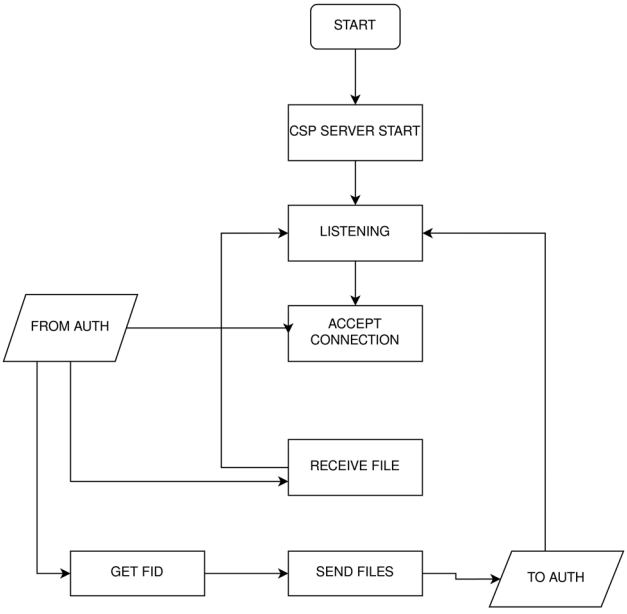

Figure 3: Workflow of CSP.

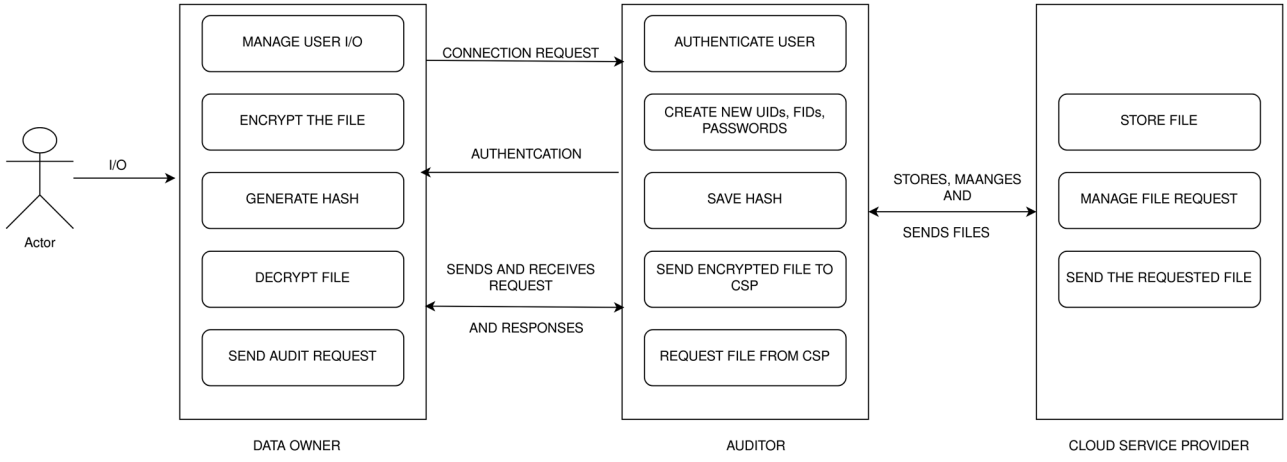

Figure 2: Architecture of the BST-based model.

4.2 Overview of BST of BST structure

This structure is the core of auditor module. As the name indicates, there are two BSTs that are connected such that the first tree us a user tree where each node of represents an individual user. The user nodes include the user's UID and IP address. It also has three other pointers, of which the first two are conventional left and right child pointers, while the third is a reference to the user's file tree as indicated by the dotted arrow in Figure 5.

Algorithm 1 shows the steps involved in generating the BST of BST structure whenever there is a new user joining the system.

Algorithm 1 Creation of user BST.

```
1:  function insertUser (userId, key)
2:      root ← insertUserNode (root, userId, key)
3:  end function
4:  function insertUserNode (root, userId, key)
5:      if root is null then
6:          return new UserNode (userId, key)
7:      end if
8:      if key < root.key then
9:          root.left ← insertUserNode (root.left, userId, key)
10:     else if key > root.key then
11:         root.right ← insertUserNode (root.right,
        userId, key)
12:     end if
13:     return root
14: end function
```

Each file uploaded to the CSP is referenced via the file tree, as shown in Figure 6. Each node represents a file and holds its FID, hash, and version number. It also contains two conventional pointers to the node's left and right children. The UIDs and FIDs determine the structure of the trees.

Algorithm 2 depicts the creation of the file BST after searching the user BST for the user

Algorithm 2 Creation of file BST

```
1 :  function searchUser (root, userId)
2 :      if root is null or root.userId = userId then
3 :          return root
4 :      end if
5 :      if userId < root.userId then
6 :          return searchUser (root.left, userId)
7 :      else
8 :          return searchUser (root.right, userId)
9 :      end if
10 : end function
11 : function insertFile (userId, fileId, versionNumber,
     timestamp)
12 :     userNode ← searchUser (root, userId)
13 :     if userNode is not null then
14 :         userNode.fileRoot ← insertFileNode
     (userNode.fileRoot, fileId, versionNumber,
     timestamp)
15 :     end if
16 : end function
17 : function insertFileNode (root, fileId,
     versionNumber, timestamp)
18 :     if root is null then
19 :         return new FileNode(fileId, versionNumber,
     timestamp)
20 :     end if
21 :     if fileId < root.fileId then
22 :         root.left ← insertFileNode (root.left, fileId,
     versionNumber, timestamp)
23 :     else if fileId > root.fileId then
24 :         root.right ← insertFileNode (root.right, fileId,
     versionNumber, timestamp)
25 :     end if
26 :     return root
27 : end function
```

To insert a new node, an ID comparison is performed, and traversal through the tree leads to the pointer where the new node must be connected. If the node being deleted is a leaf node, straightforward deletion will occur. If the child on the right does not exist, the child on the left will take its place. If the child to the left does not exist, the child to the immediate right will take its place. In all other circumstances, the node with the least value on the right child will replace the current node, and for a file update, the hash value and version of the relevant node will be replaced with the new values. Once the CSP and auditor start and go to the listening mode, the DO attempts to connect with the auditor server, as shown in Figure 7.

4.3 Process framework

After a link or a session has been established, the auditor inquires whether the user is new. If the user responds affirmatively, auditor provides the user with a UID and password; otherwise, it requests the UID and password from the user. The auditor then compares the UID and

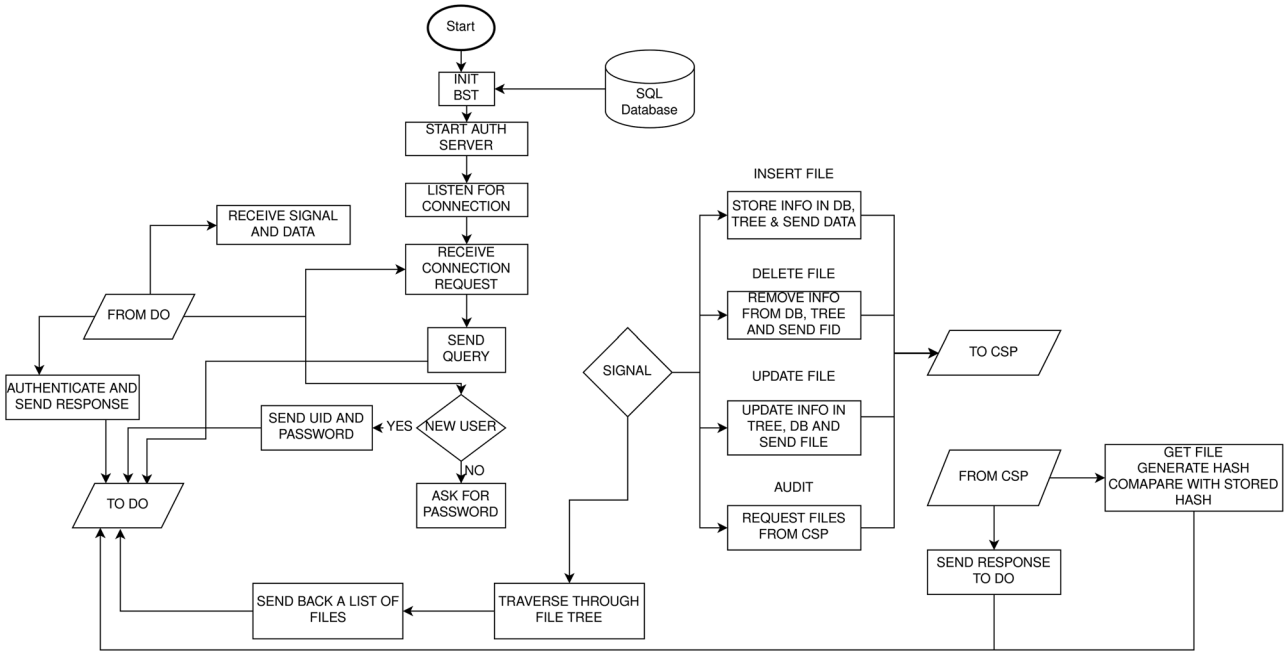

Figure 4: Workflow of auditor.

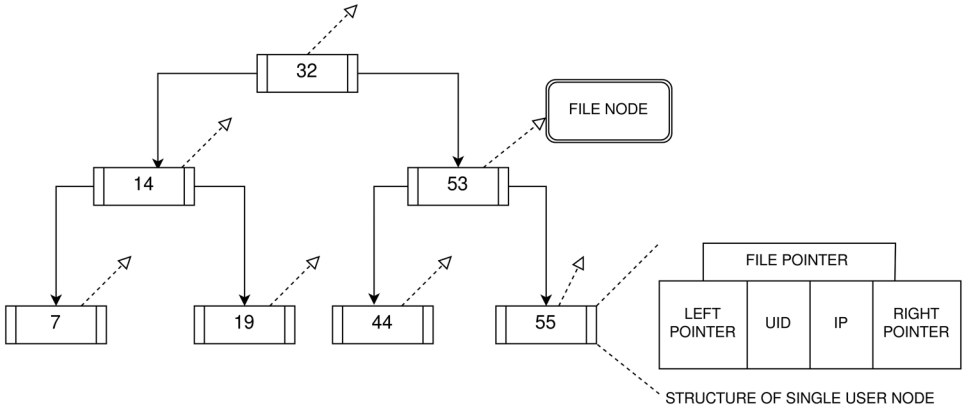

Figure 5: User tree and single BST node.

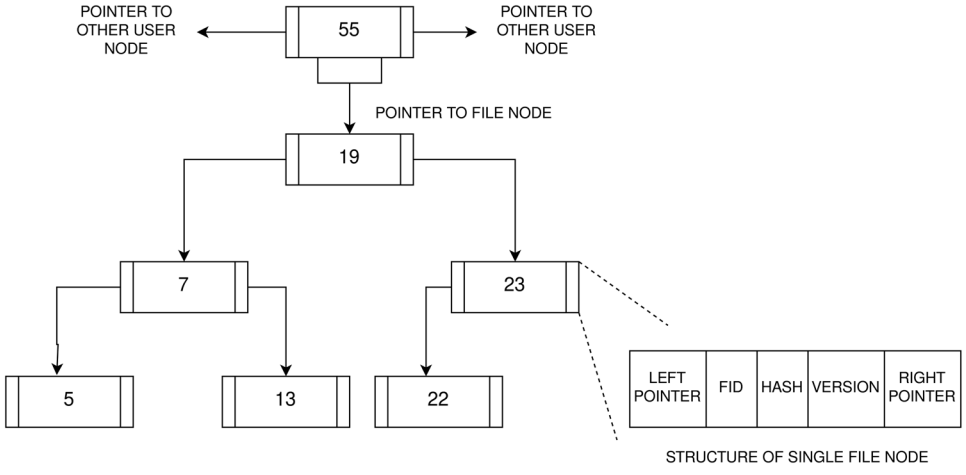

Figure 6: File tree and single file node.

password up to three times and blocks the connection from the incoming IP if the comparison fails. If the connection is successful, the user can perform the following operations or requests.

4.3.1 Insert

The insert file operation is storing the file into CSP. To perform this operation, the DO, first will encrypt the file and generate SHA-2 hash of the corresponding file. This encrypted file is then transmitted to the auditor along with the hash of the file. The auditor generates a random FID and returns the FID to the DO. It then searches the BST for the UID and then traverses through the file BST of the user to then store the received information such as hash, version, and FID of the respective user. It will create connection to the CSP and transmit the file to be saved in the CSP using the UID as the document number and FID as file number.

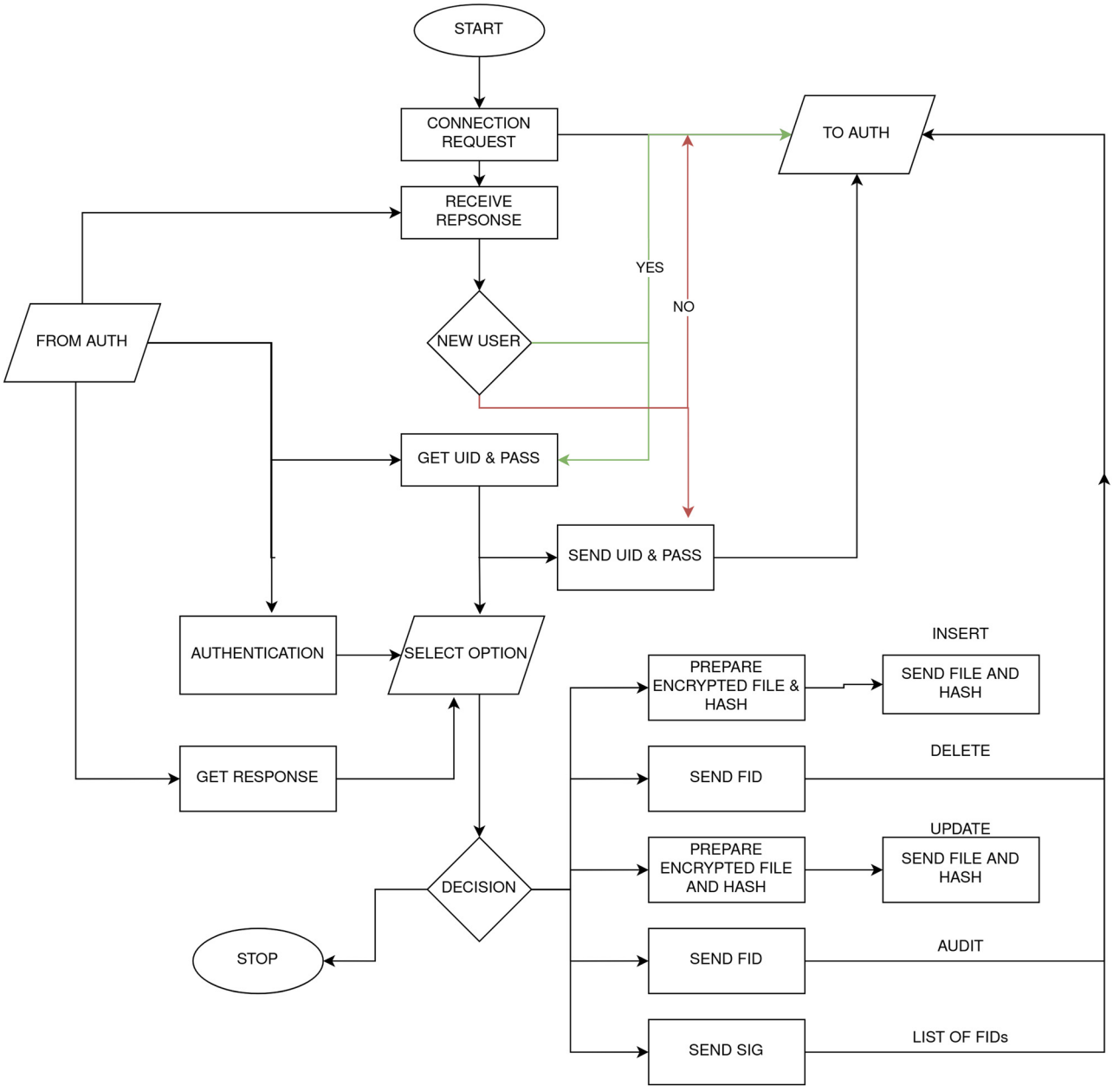

Figure 7: Workflow of DO.

4.3.2 Delete

If the DO chooses the delete operation, then DO must enter the FID and submit it to the auditor. The auditor in turn searches the user BST for the UID and then searches the file BST to delete the file's information from the tree. It also requests the CSP to remove the files from the storage.

4.3.3 Update

If the user chooses the update operation, then the updated or modified file is encrypted at the user's end and run through hash function to generate the corresponding hash. It then sends the data to the auditor along with the FID. The auditor searches for the file in the BST and then updates the metadata of the file with an increment in the version number of the file. It then sends the file to CSP where the existing file is overwritten.

4.3.4 Audit

During the auditing process, the DO will transmit the FID to be audited, while the auditor will request the file from the CSP using the UID and the FID. In the process, the generation of the challenge involves creating a set of file numbers to be verified, denoted as C = fid, $1 \le c$, $c \le n$, and selecting a random file number from the set, denoted as $S = s_i$, $i \in C$. Additionally, a random number $\rho \in Z$ is generated, and g is raised to the power of ρ, denoted as $\xi = g^{\rho}$. The cloud auditor then calculates y raised to the power of ρ, denoted as $\eta = y^{\rho}$, for use in verification. The challenge is then sent to the CSP, who generates a proof and returns the result to the cloud auditor for verification.

Once the auditor obtains the file from the CSP, it will compute the following equation to verify the proof sent from CSP:

$$\mathrm{TP} = \lambda^{\rho}.\ e(u^{H(v_i \| t_i).s_i}, v^M, \rho). \tag{3}$$

If Equation (3) is proven to be correct, then the data integrity verification process is considered to be succeeded. The verification process is in par with the file's SHA-2 hash and compare it to the hash contained in the file node in the BST. If the calculated hash matches the hash stored in the BST, then the file has not been altered since it was stored in the cloud. On the other hand, if the calculated hash does not match the hash stored in the BST, then it indicates that the file has been modified since it was last stored in the cloud, and appropriate measures can be taken to address the issue. Figure 8 describes the auditing process followed in the proposed work. Table 1 summarizes the steps involved in the auditing process.

This approach of comparing hash values ensures the integrity of the file and is a widely adopted practice for file verification in various domains. By following this process, the auditor can ensure that the data has not been tampered with, and the DO can be assured that their data remains secure and unaltered in the cloud.

On the other hand, after receiving the challenge from the cloud auditor, the CSP generates a response proof, consisting of three parts: tag proof, file proof, and auxiliary auditing proof. The tag proof, given by the equation

$$\mathrm{TP} = \prod_{i \in C} e(\sigma_i, \rho)^{s_i}, \tag{4}$$

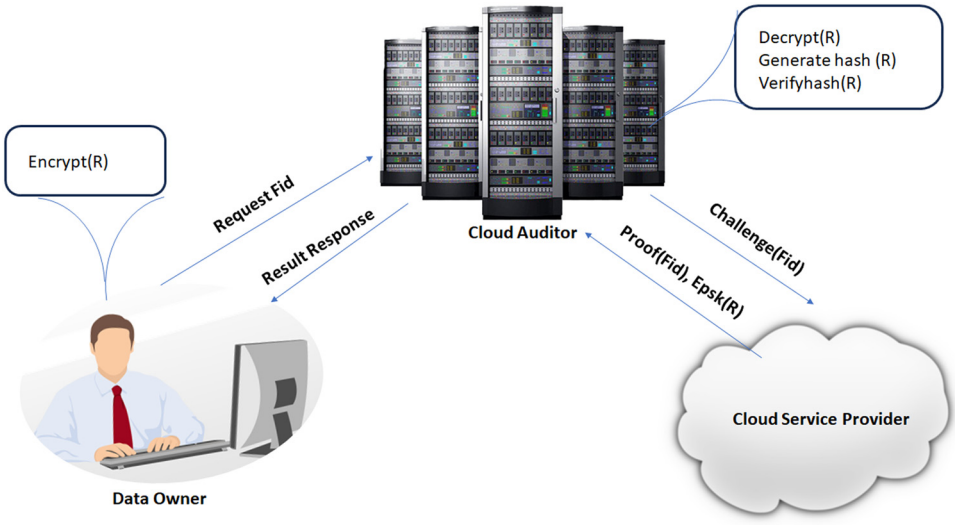

Figure 8: Steps involved in auditing process.

Table 1: Steps involved in auditing process

O:$E_{p_{sk}}$(R)
$U_1 \rightarrow$ CA: Request R_{Fid}
CA $\rightarrow$ CSP: Challenge(Fid)
CSP $\rightarrow$ CA:Proof(Fid)
CSP $\rightarrow$ CA: $E_{p_{sk}}$ (R)
CA: $D_{p_{su}}$ ($E_{p_{sk}}$(R))
CA: Generatehash(R)
CA: Verifyhash(R)
CA $\rightarrow$ U_1 : $E_{p_{sk}}$(R) with integrity verification result

is computed by taking the product of the pairing between the signature σ_i and the random number ρ, raised to the power of the corresponding element s_i in set S. The file proof, given by the equation

$$M = \sum_{i \in C} m.\ s_i + r, \tag{5}$$

is calculated by taking the sum of the product between the file m and the element s_i, for each file i in set C, and adding a random padding/mask r used to protect the data privacy. Finally, the auxiliary proof is derived using the following equation:

$$\lambda = e(v, y)^{-r}, \tag{6}$$

where v is a verification key provided by the cloud auditor, and y is the PK of the CSP. Once these values are computed, the CSP sends the tuple $\langle \text{TP}, M, \lambda \rangle$ to the cloud auditor for verification.

4.3.5 Retrieve and display file list

The user may get their saved files using the retrieve option, and when the file has been sent to the DO, the DO will decrypt the file and store it in local storage. Using the List file option, the user can see the files saved on the CSP.

5 Analysis and results obtained

This section provides the analysis of the BST-based model in terms of correctness and resistance to forging attacks. It also depicts the comparative results obtained for ECC and RSA. The section concludes with the highlight that the BST-based model outperforms the index-list based method through a statistical analysis.

5.1 Threat analysis and mitigation

This section enumerates a comprehensive list of potential threats that have been identified as potential vulnerabilities.

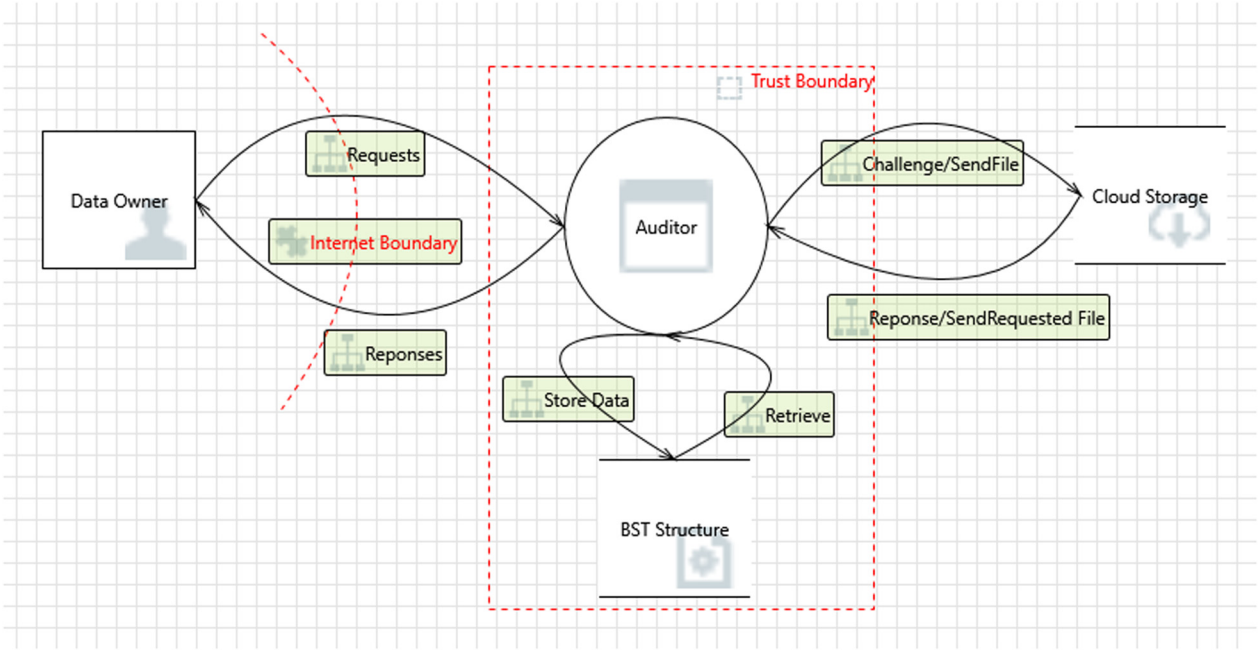

Figure 9: Generic threat model for BST-based model.

Additionally, it outlines corresponding mitigation methods for each of these threats. To analyze the possible threats, Microsoft threat modeling tool has been used. Figure 9 shows the generic threat model that has been used to identify the possible threats.

As described in the Section 4.1, there are three primitive modules in the BST-based model. The DO encrypts the data, generates the hash of the encrypted data, and sends the files to the auditor. Once the auditor generates the file number, it is sent back to the DO. Auditor has BST storage in it which stores various metadata related to the user and the respective files of the users. It stores and fetches these data to and from the BST of BST whenever the need arises. The CSP module, in turn, stores the data that the auditor sends. When the auditor sends the challenge to the CSP, it sends the necessary response back to the auditor.

Table 2 summarizes the threats that were identified along with the mitigation steps that were taken as a countermeasure to overcome those threats.

Table 3 compares the BST-based structure doubly linked list based index table (DLIT) structure with index-based method in terms of whether the methods handle the listed vulnerabilities or not.

Table 4 shows a comparison of communication cost in terms of the various operations that are involved in various state of the arts.

Table 2: Identified threats and their mitigation steps in BST-based model

Threats	Mitigations in BST-based method
Spoofing	1. Authentication mechanism has been used 2. Encryption to protect the data
Tampering	1. Appropriate authorization 2. Hashes have been used 3. Use of suitable tamper-resistant protocols
Repudiation	1. Digital signatures have been used. 2. Version numbers 3. Timestamps
Information disclosure	1. Authorization techniques 2. Encryption
Denial of service	1. Authentication 2. Authorization 3. QoS
Elevation of privilege	1. Run with least privilege
Impersonation	1. Authentication 2. Authorization
Eavesdropping	1. SSL-encrypted network traffic 2. Certificate check when connection is established between the parties

Table 3: Comparison of various state of arts with the BST of BST method

Techniques	Public auditing	Data privacy	Dynamic auditing
PDP [3]	√	×	×
PoR [14]	×	NoD	×
IHT-PA [5]	√	√	√
DAP [44]	√	√	√
DPDP [3]	×	NoD	√
DHT-PA [6]	√	√	√
DLIT-PA [7]	√	×	√
AHT-PA [30]	√	√	√
BST-based	√	√	√

√: “Supports”; ×: “Does not support”; NoD: “Not defined/no demand”.

5.2 Scheme analysis

This section confidently asserts the effectiveness of the BST-based method against forging attacks and confirms its correctness.

Correctness: The verification process of the owner’s identity as a rightful cloud user is carried out with utmost confidence via the signature scheme. Once the CSPs follow the appropriate standards, passing the auditing process becomes a guaranteed success. The file’s number, version number, name, and timestamp are all utilized to create a signature that is protected by the BLS signature against any potential tampering

When data are sent from the auditor to the CSP for verification, it is encrypted using a secret key to produce Epsk. These encrypted data contain the PK, message digest, file identifier, version number, and timestamp. If the data are modified by an adversary, the version number and message digest will change, resulting in incorrect encrypted data (Epsk*). As a result, when the auditor receives tampered data from the CSP and performs the check, it will fail if either the message digest does not match or the timestamp is different.

Table 4: Comparison of communication costs of various state of the arts

Methods	Search	Insert	Delete	Update	Audit
PDP [3]	×	×	×	×	×
PoR [14]	×	×	×	×	×
IHT-PA [5]	×	×	×	O(1)	O(c)
DAP [44]	×	×	×	×	×
DHT-PA [6]	×	×	×	O(1)	O(c)
DLIT-PA [7]	×	×	×	O(1)	O(c)
AHT-PA [30]	O(n)	O(1)	O(1)	O(1)	O(c)
BST-based	O(log n)	O(1)	O(1)	O(1)	O(c)

√: “Supports”; ×: “Does not support/not defined/no demand”.

Table 5: RSA *vs* ECC

Security (in Bits)	RSA key length required	ECC key length required
80	1,024	160–223
112	2,048	224–255
128	3,072	256–383
192	7,680	384–511
256	15,360	512+

To verify the accuracy of the CSP's response, the lambda value of the response is analyzed. This value is denoted as $\lambda^{\rho}. e(u^{H(v_i||t_i).s_i}. v^{M}, \rho)$ = TP, where TP is the tag proof. If the auxiliary proof is incorrect, the verification formula (4) will not hold. The BLS signature scheme ensures that the tag-proof TP cannot be forged.

Resistance to forging attacks: The block proof M is proven to be unforgeable, as demonstrated below. Let us assume that a false proof is submitted to the cloud auditor by the CSP (TP, M*, λ), where

$$M = \sum_{i \in C} m. s_i + r \neq M = \sum_{i \in C} m. s_i + r. \tag{7}$$

If the CSP successfully passes the verification, the following equation holds for valid proofs:

$$\begin{aligned} &\lambda^{\rho}. e(u^{H(v_i||t_i).s_i}. v^{M}, \rho) \\ &\quad = e(v, y)^{-r}. e(u^{H(v_i||t_i).s_i}. v^{\sum_{i \in C} m.s_i + r}, \rho). \end{aligned} \tag{8}$$

This equation readily refutes the presumption made in the false proof, proving that forging attacks are effectively counteracted. Furthermore, as Shen *et al.* [7] have explained, replay and replacing attacks can be effectively blocked or halted.

5.3 Performance analysis

5.3.1 RSA vs ECC

The Table 5 demonstrates that ECC provides better security than RSA, even with significantly smaller key sizes.

To demonstrate that it is a faster encryption system, both encryption systems were compared by encrypting files with 300 byte size increments in each iteration, recording the time required to complete them. The results were charted and two cases were presented: one with the same degree of security as shown in Figure 10 and another with the same key size as shown in Figure 11.

The graphs demonstrate that ECC surpasses RSA in every category, including security and performance.

5.3.2 BST of BST *vs* DLIT

This section details the space complexity and time complexity concerning the BST-based method and DLIT method.

5.3.2.1 Space complexity

The space utilization of both algorithms is directly proportional to the number of nodes present in the data structure. This is because the creation of a new node requires an allocation of memory of the size of one node class every

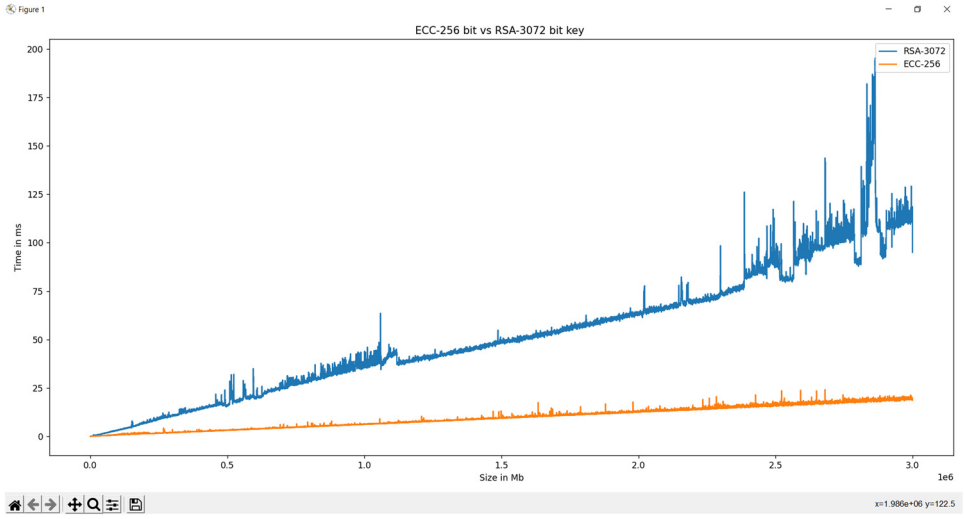

Figure 10: RSA 3072 vs ECC 256.

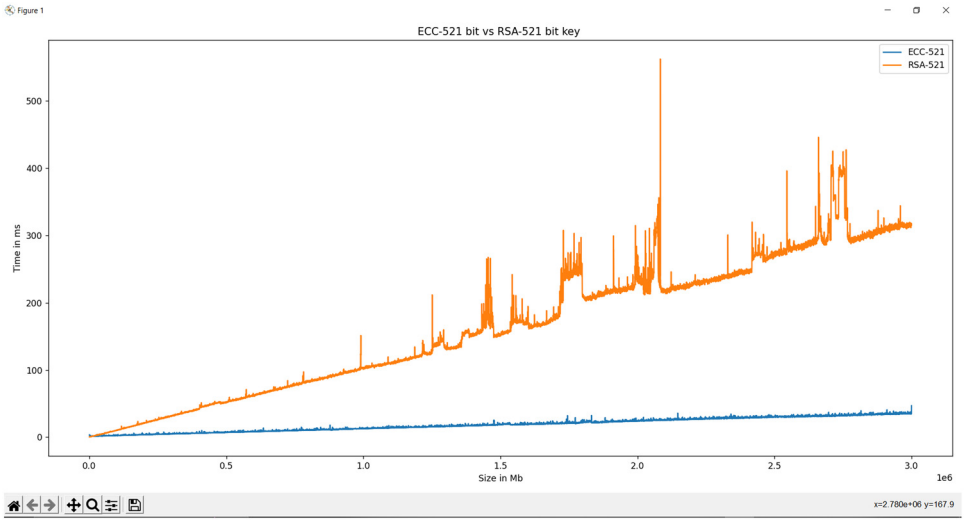

Figure 11: RSA 521 vs ECC 521.

time it is generated. As a result, at any given moment, the stack allocation of both data structures will be equivalent to the number of nodes present. Therefore, the space complexity of both algorithms can be represented by the expression $O(n)$, where n signifies the number of nodes in the data structure.

5.3.2.2 Time complexity

The algorithm for BST implements a divide-and-conquer strategy, where both the tree and search set are iteratively halved. This results in an average time complexity of $O(\log n)$. To locate a specific file node, the process involves traversing the user tree to find the corresponding user node, followed by traversing the file tree to access the desired file node. As a result of this two-step process, the overall average time complexity of the BST algorithm is estimated to be $O(\log n * \log m)$, where n represents the number of user nodes and m represents the number of file nodes.

A linked list is a data structure in which each node is connected to the next node using a pointer. To locate a specific node in a linked list, one has to traverse each subsequent node until the desired node is found. This traversal process results in a time complexity of $O(n)$, where n is the number of nodes in the list.

When searching for a file node, the process becomes even more time-consuming. The file node must be found by first browsing through a linked list of user nodes and then traversing the list of file nodes. This results in a time complexity of $O(n^2)$.

As depicted in Figure 12, the BST-based method performs better than the index-based linked list method

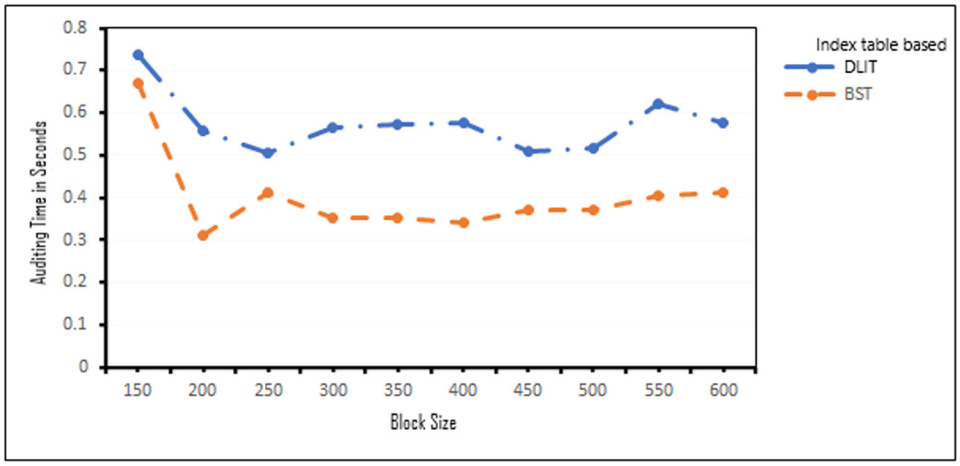

Figure 12: Comparison of index table method and BST method in seconds for auditing.

during the auditing process. The BST method has a faster time complexity, making it more efficient in this scenario.

5.4 Statistical analysis

The efficacy of the proposed methods can be scientifically evaluated through hypothesis testing, specifically analysis of variance (ANOVA). Both methods were implemented using a similar system setup and configuration to ensure fair comparison.

The hypothesis for the validation is defined as follows:

H_0 states that there is no significant difference between the index table method and the BST method,

H_1 states that there is a significant difference between the two methods.

To determine which hypothesis holds true, ANOVA was performed using the observed values for various block-sized files, as presented in Table 6. These observations were made in terms of milliseconds for the auditing process. The results of ANOVA will determine if there is a significant difference in performance between the two proposed methods.

Table 7 provides the details about the various factors that are calculated.

Table 8 details about the prerequisite value calculations for the F-calculation of ANOVA.

From the prerequisite values calculated in Table 8, the F-ratio is given by

$$F\text{-ratio} = \frac{\text{MS between}}{\text{MS within}} = 20.34781. \tag{9}$$

Therefore, $F_{\text{calculated}}$ is 20.34781. This value is then verified with the t-table where

$$F(1, 18) = 4.4139. \tag{10}$$

Table 6: Observed data in seconds

Index table method	BST method
0.738012698	0.671239259
0.555915283	0.312453451
0.503742505	0.412434441
0.563627574	0.352145123
0.573763049	0.352454631
0.576584172	0.342455512
0.509596796	0.372431981
0.517415459	0.371424214
0.621983678	0.402987652
0.575574678	0.412612431

Table 7: Calculating the factors for ANOVA

Variables	Equation	Value obtained
Total, T		9.738854587
Number of samples, n		20
Correction factor, CF	$\frac{T^2}{n}$	4.742264433
Total sum of square, SS_{Tot}	$\sum X_{ij}^2 - \frac{T^2}{n}$	0.283190861
Sum of squares between, SS_{bet}	$\sum \frac{T_j^2}{n_j} - \frac{T^2}{n}$	0.150264495
Sum of squares within, SS_{Within}	$\sum X_{ij}^2 - \sum \frac{T_j^2}{n_j}$	0.132926366

Comparing the calculated and tabulated value of t-statistic:

$$F_{\text{calculated}} > F_{\text{tabulated}}. \tag{11}$$

The results of Equations (1) and (2) were substituted into Equation (3), revealing that the calculated value of F ($F_{\text{calculated}}$) was higher than the tabulated value of F ($F_{\text{tabulated}}$). This indicates that the calculated value falls within the rejection region; hence, the results reject the null hypothesis with the significance level that falls below 5%. This rejection of the null hypothesis suggests that there is a significant improvement in the execution time of the data integrity check process when using the BST method.

In other words, the analysis demonstrates that the BST method leads to a marked reduction in the execution time required to perform the data integrity check process, compared to traditional methods. This result supports the effectiveness of the BST method in improving the efficiency of data integrity checks in cloud environments.

6 Conclusion and future scope

6.1 Conclusion

In conclusion, the proposed study has focused on optimizing data retrieval processes to bolster data integrity verification in cloud environments. With the exponential

Table 8: Prerequisite values to calculate $F_{\text{calculated}}$

Source of variation	SS	d.f	MS
Between sample	143.937496	(2 – 1) = 1	0.150264
Within sample	11.4724	(20 – 2) = 18	0.007385
Total	0.283190861	(20 – 1) = 19	

growth of global data transfer, cloud data storage has become a pivotal solution, necessitating robust mechanisms for maintaining data integrity.

The proposed methodology, centered around a data integrity auditor framework, emphasizes efficient data retrieval while ensuring enhanced data integrity verification. By leveraging a BST for streamlined metadata management and employing ECC for secure data encryption, our approach optimizes data retrieval processes crucial for integrity verification.

Our comparative analysis against established methods such RSA and index table-based DLIT showcases the efficiency and security benefits of our proposed solution. These findings underscore the significance of optimizing data retrieval mechanisms to fortify data integrity verification in cloud environments.

Furthermore, it contributes to the ongoing discourse on cloud data management by providing a comprehensive approach that addresses critical challenges in cloud-based data storage and security. By optimizing data retrieval processes, we not only enhance data integrity verification but also pave the way for more efficient and secure cloud data management practices.

Overall, the work underscores the importance of considering data retrieval optimization as a key aspect of ensuring data integrity in cloud environments, highlighting avenues for further research and development in this domain.

6.2 Future scope

This article provides a foundation for future work that can be done on this model and system such as:

- Enhancing the security measures to provide an even higher level of protection using Curve448 or Ed448.
- Enhancing the security measures of the hardwares used in the system by testing for SIKE, Kyber on Cortex M4 or ARM processors.
- Various fault detection techniques or error detection methods for block ciphers such as Camilla, Midori cipher, Qualcomm ARM Authenticator, or a stream cipher such as WAGE.

These improvements hold the promise of further advancing the capabilities and potential of the model and system presented in this work.

Acknowledgement: The authors would like to thank Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, for providing the required resources to conduct this experiment.

Funding information: The authors state no funding involved.

Author contributions: All authors have accepted responsibility for the entire content of this manuscript and consented to its submission to the journal, reviewed all the results, and approved the final version of the manuscript. AKC designed the experiments, conceptualized the models, performed the field study, curated the data, prepared the original draft of the article and validated. BM performed the visualization, investigation, reviewed and edited the original article and validated. VP contributed in realizing the methodology, writing the article and validation.

Conflict of interest: The authors state no conflict of interest.

Data availability statement: Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study. All data generated or analyzed during this study are included in this published article.

References

[1] Gartner. Gartner Forecasts Worldwide Public Cloud End-UserSpending to Grow 23% in 2021; [Cited 02 May 2022]. https://www.gartner.com/en/newsroom/press-releases/2021-04-2-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021.

[2] egnyte.com Data Auditing. [Cited 26 April 2021]. https://www.egnyte.com/guides/governance/data-auditing.

[3] Erway CC, Küpçü A, Papamanthou C, Tamassia R. Dynamic provable data possession. ACM Trans Inform Syst Security. 2015 Apr;17(4):1–29. doi: 10.1145/2699909.

[4] Wang Q, Wang C, Ren K, Lou W, Li J. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distributed Syst. 2011;22(5):847–59.

[5] Zhu Y, Ahn G, Hu H, Yau SS, An HG, Hu C. Dynamic audit services for outsourced storages in clouds. IEEE Trans Services Comput. 2013;6(2):227–38.

[6] Tian H, Chen Y, Chang C, Jiang H, Huang Y, Chen Y, et al. Dynamic-Hash-Table based public auditing for secure cloud storage. IEEE Trans Services Comput. 2017;10(5):701–14.

[7] Shen J, Shen J, Chen X, Huang X, Susilo W. An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Trans Inform Forensics Security. 2017;12(10):2402–15.

[8] S. R. Department. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025. [Cited March 2022]. https://www.statista.com/statistics/871513/worldwide-data-created/.

[9] Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, et al. Provable data possession at untrusted stores. Proceedings of the 14th ACM Conference on Computer and Communications Security; 2007. p. 598–609.

[10] Juels A, Kaliski BS. PORs: Proofs of retrievability for large files. Proceedings of 14th ACM Conference on Computer and Communications Security (CCS '07); 2007 November. p. 584–97.

[11] Liu D, Zic J. Proofs of Encrypted Data Retrievability with Probabilistic and Homomorphic Message Authenticators. In: Society IC, editor. IEEE Trustcom/BigDataSE/ISPA; 2015.
[12] Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. In: International Conference on the Theory and Aapplication of Cryptology and Information Security. Springer; 2001. p. 514–32.
[13] Luo X, Zhou Z, Zhong L, Mao J, Chen C. An effective integrity verification scheme of cloud data based on BLS signature. Wiley Online Library; 2018. doi: 10.1155/2018/2615249.
[14] Du R, Deng L, Chen J, He K, Zheng M. Proofs of Ownership and Retrievability in Cloud Storage. In: Society IC, editor. IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications; 2014. p. 328–335.
[15] Kwon O, Koo D, Shin Y, Yoon H. A secure and efficient audit mechanism for dynamic shared data in cloud storage. The Scientific World. Sci World J. 2014;2014:820391.
[16] More S, Chaudhari S. Third party public auditing scheme for cloud storage. Procedia Computer Sci. 2016;79:69–76.
[17] Shah H, Shah J, Desai U. Third Party Public Auditing Scheme for Security in Cloud Storage. Int J Trend Scientif Res Development (ijtsrd). 2019 April;3:179–84.
[18] Yuan J, Yu S. Public integrity auditing for dynamic data sharing with multiuser modification. IEEE Trans Inform Forensics Security. 2015 August;10(8):1717–26.
[19] Zargad SV, Tambile AV, Sankoli SS, Bhongale RC. Data Integrity Checking Protocol with Data Dynamics and Public Verifiability for Secure Cloud Computing. Int J Comput Sci Inform Tech (IJCSIT). 2014;5(3):4062–4.
[20] Kaaniche N. Cloud Data Storage Security Based on Cryptographic Mechanisms [dissertation]. Informatique, Télécommunications et Électronique de Paris; 2014.
[21] Li R, Shen C, He H, Gu X, Xu Z, Xu CZ. A lightweight secure data sharing scheme for mobile cloud computing. IEEE Trans Cloud Comput. 2017 Jan 6;6(2):344–57.
[22] Anisetti M, Ardagna CA, Damiani E, Gaudenzi F. A semi-automatic and trustworthy scheme for continuous cloud service certification. IEEE Trans Services Comput. 2017 Jan 24;13(1):30–43.
[23] Sarkar MK, Chatterjee T. Enhancing data storage security in cloud computing through steganography. Int J Netw Security. 2014 Jan 1;5(1):13.
[24] Sun L, Xu C, Zhang Y, Chen K. An efficient iO-based data integrity verification scheme for cloud storage. Sci China Inform Sci. 2019 May 1;62(5):59101.
[25] Wang C, Di X. Research on integrity check method of cloud storage multi-copy data based on multi-agent. IEEE Access. 2020 Jan 15;8:17170–8.
[26] Ganesh SM, Manikandan SP. An efficient integrity verification and authentication scheme over the remote data in the public clouds for mobile users. Security Commun Netw. 2020;2020(1):9809874.
[27] Chidambaram N, Raj P, Thenmozhi K, Amirtharajan R. Enhancing the security of customer data in cloud environments using a novel digital fingerprinting technique. Int J Digital Multimedia Broadcast. 2016;2016(1):8789397.
[28] Zhong W, Liu Z. Efficient proof of ownership for cloud storage systems. Guangzhou, China: School of Computer Science and Technology, Guangdong University of Technology; 2017. https://aip.scitation.org/doi/abs/10.1063/1.4992867.
[29] Canto AC, Kermani MM, Azarderakhsh R. Reliable constructions for the key generator of code-based post-quantum cryptosystems on FPGA. ACM J Emerg Tech Comput Syst. 2022 Dec 9;19(1):1–20.
[30] Chen W, Tian H, Chang CC, Nan F, Lu J. Adjacency-?hash-?table based public auditing for data integrity in mobile cloud computing. Wireless Commun Mobile Comput. 2018;2018(1):3471312.
[31] Azarderakhsh R, Koziel B, Jalali A, Kermani MM, Jao D. NEON-SIDH: efficient implementation of supersingular isogeny Diffie–Hellman Key-exchange protocol on ARM. IACR Cryptol ePrint Arch. 2016;2016:669.
[32] Kermani MM, Azarderakhsh R, Mirakhorli M. Multidisciplinary approaches and challenges in integrating emerging medical devices security research and education. In 2016 ASEE Annual Conference & Exposition. 2016 Jun 26.
[33] Kermani MM. Fault detection schemes for high performance vlsi implementations of the Advanced Encryption Standard [dissertation], Faculty of Graduate Studies, University of Western Ontario; 2007.
[34] Yan YX, Wu L, Xu WY, Wang H, Liu ZM. Integrity audit of shared cloud data with identity tracking. Security Commun Netw. 2019;2019(1):1354346.
[35] Chandel S, Cao W, Sun Z, Yang J, Zhang B, Ni TY. A multi-dimensional adversary analysis of RSA and ECC in blockchain encryption. In Advances in Information and Communication: Proceedings of the 2019 Future of Information and Communication Conference (FICC). Springer; 2020. p. 988–1003.
[36] Niasar MB, Azarderakhsh R, Kermani MM. Optimized architectures for elliptic curve cryptography over Curve448. Cryptology ePrint Archive. 2020.
[37] Cintas-Canto A, Kermani MM, Azarderakhsh R. Reliable architectures for finite field multipliers using cyclic codes on FPGA utilized in classic and post-quantum cryptography. IEEE Trans Very Large Scale Integration (VLSI) Syst. 2022;31(1):157–61.
[38] Canto AC, Kermani MM, Azarderakhsh R. CRC-based error detection constructions for FLT and ITA finite field inversions over GF (2m). IEEE Trans Very Large Scale Integration (VLSI) Syst. 2021 Mar 10;29(5):1033–7.
[39] Canto AC, Sarker A, Kaur J, Kermani MM, Azarderakhsh R. Error detection schemes assessed on FPGA for multipliers in lattice-based key encapsulation mechanisms in post-quantum cryptography. IEEE Trans Emerg Topics Comput. 2022;11(3):791–7.
[40] Kaur J, Canto AC, Kermani MM, Azarderakhsh R. Hardware constructions for error detection in WG-29 stream Cipher benchmarked on FPGA. IEEE Trans on Computer-Aided Design of Integrated Circuits and Syst. 2024;43(4):1307–11.
[41] Mozaffari-Kermani M, Azarderakhsh R, Ren K, Beuchat JL. Guest editorial: introduction to the special section on emerging security trends for biomedical computations, devices, and infrastructures. IEEE/ACM Trans Comput Biol Bioinform. 2016 May 1;13(3):399–400.
[42] Karam RA, Katkoori S, Kermani MM. Work-in-progress: Hyflex hands-on hardware security education during covid-19. In 2022 IEEE World Engineering Education Conference (EDUNINE), IEEE; 2022 Mar 13. p. 1–4.
[43] Kermani MM, Bayat-Sarmadi S, Ackie AB, Azarderakhsh R. High-performance fault diagnosis schemes for efficient hash algorithm blake. In 2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS), IEEE; 2019 Feb 24. p. 201–204.
[44] Yang K, Jia X. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parallel Distributed Syst. 2012 Sep 24;24(9):1717–26.