Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2025.058717

ARTICLE

Adaptive Attribute-Based Honey Encryption: A Novel Solution for Cloud Data Security

Reshma Siyal[1], Muhammad Asim[2,*], Long Jun[1], Mohammed Elaffendi[2], Sundas Iftikhar[3], Rana Alnashwan[4] and Samia Allaoua Chelloug[4,*]

[1]School of Computer Science and Engineering, Central South University, Changsha, 410083, China

[2]EIAS LAB, College of Computer and Information Sciences, and Center of Excellence in Quantum and Intelligent Computing, Prince Sultan University, Riyadh, 11586, Saudi Arabia

[3]Gianforte School of Computing, Montana State University, Bozeman, MT 59717, USA

[4]Department of Information Technology, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh, 11671, Saudi Arabia

*Corresponding Authors: Muhammad Asim. Email: masim@psu.edu.sa; Samia Allaoua Chelloug. Email: sachelloug@pnu.edu.sa

Received: 19 September 2024 Accepted: 21 November 2024 Published: 17 February 2025

ABSTRACT

A basic procedure for transforming readable data into encoded forms is encryption, which ensures security when the right decryption keys are used. Hadoop is susceptible to possible cyber-attacks because it lacks built-in security measures, even though it can effectively handle and store enormous datasets using the Hadoop Distributed File System (HDFS). The increasing number of data breaches emphasizes how urgently creative encryption techniques are needed in cloud-based big data settings. This paper presents Adaptive Attribute-Based Honey Encryption (AABHE), a state-of-the-art technique that combines honey encryption with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to provide improved data security. Even if intercepted, AABHE makes sure that sensitive data cannot be accessed by unauthorized parties. With a focus on protecting huge files in HDFS, the suggested approach achieves 98% security robustness and 95% encryption efficiency, outperforming other encryption methods including Ciphertext-Policy Attribute-Based Encryption (CP-ABE), Key-Policy Attribute-Based Encryption (KB-ABE), and Advanced Encryption Standard combined with Attribute-Based Encryption (AES+ABE). By fixing Hadoop's security flaws, AABHE fortifies its protections against data breaches and enhances Hadoop's dependability as a platform for processing and storing massive amounts of data.

KEYWORDS

Cybersecurity; data security; cloud storage; hadoop encryption and decryption; privacy protection; attribute-based honey encryption

1 Introduction

In today's digital environment, online communication will soon become the primary means of communication for everything. Making efficient use of the internet simplifies our lives. These days, sharing information online comes with many security threats and difficulties [1,2]. Although encryption is crucial for data security, conventional techniques might not be strong enough to fend off sophisticated attacks. According to International Data Corporation (IDC) projections, there will be 41.6 billion Internet of Things (IoT) devices by 2025, producing around 79.4 zettabytes (ZB) of data. This significant rise in the volume and sensitivity of data emphasizes how urgently new security solutions are required. Because cloud storage providers provide dependable, scalable, and efficient data centers, they are essential for enabling smart city apps and services.

These services are essential for hosting, development, and data management for smart city administrators and residents. Furthermore, the pay-as-you-go model has improved the operational flexibility and efficiency of traditional businesses by enticing them to migrate to the cloud [3,4]. The United Nations Digital Economy Report 2019 highlights the digital economy's critical role in global economic growth, accounting for between 4.5% and 15.5% of the world GDP. This underscores the importance of continuous digital transformation and network improvements. Because cloud computing integrates technologies like big data, artificial intelligence, the Internet, and other economic systems, it is essential to the modern financial ecosystem. According to [5], the public cloud industry is predicted to increase significantly, rising by 17% from $227.8 billion in 2019 to $266.4 billion in 2020. Despite these advantages, cloud storage still faces critical security challenges.

Innovative methods such as Attribute-Based Honey Encryption (ABHE) [6] are necessary since traditional encryption approaches are frequently insufficient to counter sophisticated attacks. The capacity of ABHE to produce false ciphertexts that imitate authentic ones is noteworthy as it can mislead attackers and improve security by adding more levels of deception. This technique enables exact control over access to data stored in the cloud by enabling encryption depending on specified criteria.

Although they are strong, current cloud data security methods like CP-ABE and AES sometimes have trouble scaling and adapting to the kind of massive, dynamic datasets that are typical in Hadoop systems. These techniques lack defenses against attackers attempting unauthorized decryption and are vulnerable to both side-channel and brute-force assaults. Additionally, significant encryption overhead can slow down processing rates that are essential in settings like Hadoop which handle enormous volumes of data, making it difficult for encryption models incorporated into big data frameworks to strike a balance between security and computational efficiency.

Two main weaknesses in the existing cloud security models are addressed by this study: (1) the requirement for a multi-layered security strategy that incorporates deception techniques, such as honey encryption, to mislead attackers, and (2) the capacity to protect big data files with little effect on system performance. Large-scale data processing is made possible without sacrificing data security thanks to the AABHE paradigm, which combines CP-ABE with honey encryption principles to provide excellent security and efficiency.

Honey Encryption

To strengthen the security of encrypted data against brute-force assaults, Juels et al. [7] invented honey encryption (HE), a sophisticated cryptographic approach, in 2014. The main breakthrough of honey encryption is its capacity to produce believable decoy ciphertexts, or "honeywords," which trick adversaries into thinking they have cracked the data.

Important Honey Encryption Principles:

- Attribute-Dependent Decoys: Honey encryption can generate decoys according to particular data qualities. Because of this, the created honeywords can better fit the context of the material under protection, increasing their plausibility and usefulness as a diversion.

- Decoy Generation: Honey encryption creates several ciphertexts that mimic legitimate outputs instead of a single ciphertext that may be decrypted to disclose information. The fact that these spoofs are indistinguishable from the real data makes things much more difficult for would-be attackers.
- Enhanced Security: Honey encryption successfully thwarts brute-force assaults by offering many believable outputs. Attackers are tricked into thinking they have successfully decrypted the data when, in reality, they have only been able to access one of the ruses.
- Flexibility in Key Management: Honey encryption improves the overall security framework without requiring major modifications to current infrastructures by adapting to a variety of encryption techniques, including those used in cloud storage and secure communication systems.
- Applications for honey encryption may be found in many different fields, such as secure cloud storage, where private information has to be shielded from prying eyes. Through the integration of honey encryption with attribute-based encryption frameworks, such as AABHE, entities may offer an extra security measure that guarantees the confidentiality and integrity of their information.

The goal of this work is to provide an improved version of the Cipher Key Adaptive ABHE (AABHE) algorithm specifically for cloud storage settings. The AABHE algorithm's performance was assessed in several areas, such as usability, attack resistance, and computing efficiency. The paper offers a thorough assessment of the AABHE algorithm to facilitate its integration into cloud storage systems, to deliver a workable option for enhancing cloud data security.

The unique contributions of this paper are:

1) **Development of AABHE:** This paper presents AABHE, an encryption technique that combines honey encryption and attribute-based access control in a novel way. This methodology significantly improves security over ordinary attribute-based encryption alone by producing believable decoy data (sometimes known as "honey words") for unwanted users.
2) **Comparative Analysis:** AABHE's advantage in terms of security robustness and encryption efficiency is demonstrated by a comparative assessment with other current techniques, such as CP-ABE, KB-ABE, and AES+ABE. It achieves 98% robustness and 95% efficiency, making it appropriate for huge data files in Hadoop.
3) **Enhanced Data Handling in Hadoop:** By enabling safe, effective encryption and decryption of massive files, AABHE's architecture is personalized for Hadoop's HDFS, strengthening Hadoop against data breaches and establishing it as a more secure framework for cloud-based big data processing and storage.

The remaining portions of this paper are as follows:

Section 2 gives a summary of earlier research on cloud data security. The new AABHE-based encryption method is explained in Section 3. For different text file sizes (32, 64, 128, 256, and 512 MB), this method is compared to CP-ABE and KP-ABE algorithms in Section 4, which also explains how to integrate it into the Hadoop system. The results are thoroughly discussed in Section 5. Section 6 concludes the research. Enumerate the past research on cloud data security that has been carried out in Section 6.

2 Related Work

To protect password-based encryption, which is vulnerable to brute-force assaults, Juels et al. [7] first applied HE. To thwart attempts to guess passwords or encryption keys, the approach generates fake outputs that seem legitimate to an attacker. HE adds another degree of deception to encrypted communications during their decryption, extending this idea to secure messaging systems.

In order to safeguard massive data transferred throughout blockchain nodes, Siyal et al. [8] utilized HE. Their technique guaranteed that unauthorized users would be presented with decoy data but authorized users could still get the actual information by merging HE with Deep Siamese Neural Networks (DSNN) for key generation and blockchain for transparency. This rendered HE a practical instrument for safeguarding decentralized systems and distributed ledger technology (DLTs).

In order to improve security by making sure that the decryption results cannot be altered, a study by Cai et al. [9] suggests a blockchain-based ABE framework that integrates zero-knowledge proof techniques for verifiable outsourced decryption. This framework places a strong emphasis on efficiency and decryption operation verification, both of which are essential for cloud and blockchain systems, particularly when handling sensitive data.

Li et al. [10] expanded on this idea by combining HE with outsourced ABE decoding. This made it possible to outsource labor-intensive decryption activities to the cloud while guaranteeing that illicit efforts to decrypt data produced spoofs rather than actual data.

Using Honey Bee Behavior to Generate Realistic Decoys: Principal Participant: To enhance the creation of dummy images [11], this study combines Honey Encryption with a Honey Bee Behavior Model. Because it produces realistic decoys that make it harder for attackers to discern between actual and false data, this approach improves access control systems in cloud settings. The study demonstrates how biological concepts might be used in cryptography methods to improve security.

Using Honey Encryption for Password-Based Encryption, Bethencourt et al.'s [12] study offers a comprehensive examination of the difficulties and restrictions associated with Honey Encryption, particularly about password-based encryption (PBE). While providing ways to enhance HE's effectiveness in managing non-uniform message distributions, including passwords and credit card numbers, the research also tackles the scalability and performance difficulties associated with HE.

Al-Rafidain et al. [13] investigate the use of honey encryption in Wireless Sensor Networks (WSN), with an emphasis on enhancing data security through the combination of source and channel encryption. To safeguard data against brute-force assaults, the suggested solution uses Gaussian Frequency Shift Keying (GFSK) and Honey Encryption to encrypt information bits.

Only those meeting the access policy criteria can decrypt the data. Lewko et al. [14] explained that KP-ABE's access policies are logical expressions of attributes. For instance, an access policy might require both "employee" and "manager" attributes, necessitating that users have a private key fulfilling these conditions.

KP-ABE is employed in several contexts, including cloud computing systems and safe data exchange in the healthcare industry [5]. It provides an adaptable and efficient method of managing data access. Sahai and Waters first suggested KP-ABE in 2005 [13].

However, it also has limitations. If the access policy is not carefully constructed, it can be vulnerable to attacks and may not be suitable for all data-sharing scenarios [15–17]. Each ciphertext in KP-ABE has its attributes, with the user's private key linked to the access policy governing those attributes [18].

Ciphertext-policy attribute-based Encryption (CP-ABE) is another attribute-based encryption approach used for data stored in the cloud. CP-ABE embeds access policies within the ciphertext itself, controlling access based on user attributes [19]. For example, a policy might require the attribute "age greater than 18" for decryption. CP-ABE is useful in scenarios requiring strict access control, such as healthcare, where only authorized personnel can access sensitive data.

CP-ABE, however, is complex and computationally intensive, requiring significant resources. Due to the scheme's complexity, key management can also be challenging. Bethencourt et al. developed the first CP-ABE model [12]. Despite its flexibility, CP-ABE can face issues such as collusion attacks, where unauthorized users collaborate to gain access, and attribute revocation problems, where users retain access they should no longer have due to outdated policies [6].

Revocation is critical in ABE, requiring secure mechanisms to prevent unauthorized access. Indirect revocation schemes involve regularly updating decryption keys for non-revoked users, while direct schemes update keys in real-time. This study [20] presents a hybrid strategy that combines ABE and Monarch Butterfly Optimization (MBO). MBO, an optimization technique based on the migration of monarch butterflies, is used to choose the best encryption settings to increase system security and performance. The technique concentrates on resolving issues with cloud settings, namely safe data retrieval and storage. Using this method, the authors want to ensure effective data exchange between various cloud platforms by lowering encryption and decryption times while upholding high-security standards. Hohenberger et al. [21] developed a registered attribute-based encryption system offering fine-grained access control but requiring complex, non-black-box techniques.

Shao et al. [22] designed the scheme An Efficient Attribute-Based Encryption Scheme with Data Security Classification in the Multi-Cloud Environment." In multi-cloud contexts, it introduces a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) method that incorporates data security classification. By utilizing external decryption and encryption methods, the model is made for effective decryption with minimal processing overhead.

Kumar et al. [23] introduced a hybrid model combining the Advanced Encryption Standard (AES) and ABE algorithms, enhancing security and efficiency for cloud data protection. Ge et al. [24] proposed an attribute encryption system with steadfast decryption outsourcing, utilizing blockchain-based smart contracts to verify decryption accuracy.

Overall, KP-ABE, CP-ABE, and hybrid models like AES-ABE each offer unique benefits and challenges for securing cloud-based data. Selecting the appropriate encryption scheme requires careful consideration of the specific use case and requirements to ensure optimal security and privacy.

Table 1 presents the suggested AABHE model along with a comparative summary of current encryption techniques. Strong security protections are provided by AES and CP-ABE, but their ability to scale well with big datasets and stop unwanted access without requiring resource-intensive processing is limited. By combining attribute-based encryption with honey encryption, AABHE, on the other hand, overcomes these drawbacks and guarantees great security and efficiency while managing big files in Hadoop.

Table 1: Details such as methodology, advantages, and limitations of a few previous works

Authors	Methodology	Advantages	Limitations
Bethencourt et al. [12]	KP-ABE (Key-Policy Attribute-Based Encryption)	Fine-grained access control	Vulnerable to collusion attacks, complex key man-agreement
Hohenberger et al. [21]	Registered attribute-based encryption	Fine-grained access control	Complex, heavy non-black-box techniques
Kumar et al. [23]	Hybrid model combining AES and ABE	High level of security, efficient for cloud data	Complex implementation
Ge et al. [24]	Attribute encryption with steadfast decryption out-sourcing	Combines blockchain for secure decryption verification	Complex, requires reliable cloud service provider
Wang et al. [25]	ABHE for IoT security	Enhances IoT security, flexible access control	High computational over-head
Zhang et al. [26]	Performance analysis of ABHE in cloud storage	Improved performance metrics, efficient for large data	Requires significant computational resources
Jiang et al. [27]	Hybrid encryption scheme for secure data sharing in cloud computing.	Offers both the efficiency of symmetric encryption and the security of asymmetric encryption.	The complexity increases when managing keys and decrypting large datasets
Doe et al. [28]	Attribute-based encryption with machine learning	Combines ABE with ML for predictive security	High computational cost, complex integration
Goyal et al. [29]	Fuzzy identity-based encryption	Basis for ABE, flexible	Not scalable
Proposed AABHE	**Integrates CP-ABE with honey encryption**	**High security and efficiency (98% robustness, 95% efficiency); deception layer prevents unauthorized access**	**Higher initial setup complexity; additional resource requirements in large-scale systems**

3 System Model and Technical Routes

A major improvement over conventional honey encryption approaches is the suggested method for data encryption. Our approach uses two levels of security to overcome the drawbacks of traditional encryption techniques. A 128-bit/256-bit encryption method is used by this novel algorithm, known as Adaptive Attribute-Based Honey Encryption (AABHE), to offer an extra degree of security. Wang et al. [25] explain how to use ABHE to improve security in the Internet of Things (IoT). The writers stress the importance of its access control's flexibility in IoT contexts, where a variety of devices and user roles are present. They do point out that the system has a significant processing overhead, which might be problematic in IoT applications when resources are limited. Zhang et al. [26] examine ABHE's performance in cloud storage frameworks and show enhanced metrics and efficiency concerning managing big datasets. According to their research, ABHE can improve data security in cloud settings; nevertheless, the implementation requires a large amount of processing power, which raises questions regarding operational viability and scalability.

A hybrid encryption system is proposed by Jiang et al. [27] to improve the security of data sharing in cloud computing settings. The technique ensures both quick encryption and high levels of data protection by fusing the effectiveness of symmetric encryption with the strong security offered by asymmetric encryption. The authors stress the advantages of this strategy for protecting private data while preserving cloud system functionality. But they also point out that the difficulty of decrypting big datasets and the intricacy of key management might raise the computational cost, which could be a drawback for systems that need to be highly scalable. Although important administrative and efficiency trade-offs must be addressed in its actual implementation, this hybrid paradigm is an excellent way to secure cloud-based data exchange.

Doe et al. [28] investigate a unique way to combine machine learning with ABHE. Their method enhances security measures dynamically by combining the advantages of machine learning and ABE. The enormous computational costs of this integration and the difficulties presented by complex system integration, however, are issues that the research brings up.

The viability of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) for data security has been investigated in earlier works, including those by Goyal et al. [29]. Further, other techniques have been offered by Shobha et al. [30]. By combining system attributes with the user's private key and access rules, AABHE improves security. A user can decrypt the ciphertext [31,32] if their characteristics match the requirements stated in the encryption algorithms. The first step in the encryption process is to choose a group of attributes from the file that has to be encrypted. Next, rules defining these attributes are established. After that, the file is encrypted under these principles and password-protected for further protection. By using the concepts of honey encryption, a password may be entered to produce a string of phrases that are meant to trick possible attackers.

Several important publications evaluate the vulnerabilities and protection methods against Distributed Denial of Service (DDoS) attacks in cloud settings and Software-Defined Networks (SDN) and help to comprehend and mitigate these threats. Badotra et al. [33] emphasize how vulnerable SDN controllers are to DDoS assaults, focusing in particular on OpenDaylight (ODL) and Open Networking Operating System (ONOS). They highlight the necessity of more robust security measures in distributed cloud configurations by simulating DDoS attacks in cloud systems and using real-time monitoring tools like Wireshark to identify fraudulent traffic. However because their study is restricted to particular situations, it might not cover all possible DDoS attack paths. Similarly, an integrated SDN framework utilizing machine learning approaches for early DDoS detection is proposed by Najafimehr et al. [34]. Their approach achieves over 90% detection accuracy by filtering malicious traffic using Recursive Feature Elimination (RFE) and event correlation. Despite its effectiveness, this system might not be able to handle real-time processing in high-throughput settings. A more comprehensive analysis of public datasets and different DDoS mitigation techniques is given in another paper. The significance of datasets like CICIDS2017 for creating detection models with higher accuracy is emphasized, while machine learning-based defensive frameworks like Pro-defensive are highlighted. This study highlights the dearth of comprehensive datasets that cover a wide range of attack types, but it also shows that integrating datasets and models might further improve the resilience of SDN systems against developing DDoS attacks [35].

Usama et al. [36] highlight how crucial it is to have strong security measures in place in multi-cloud systems, especially for applications related to conveyance. The Blowfish algorithm is presented by Usama et al. as an efficient encryption technique that is successful in protecting data on various cloud platforms. According to the results, using many cloud environments can improve data security while maintaining adherence to security guidelines. Because it emphasizes the necessity of flexible security frameworks that can function well in a variety of cloud environments, this study is especially pertinent to my research.

With an emphasis on trust management via a trusted cloud broker [37], this article presents a three-phase methodology for managing service level agreements (SLAs) in cloud settings. To guarantee that cloud services fulfill specified security and performance requirements, Aftab Syed et al. tackle the difficulties of service monitoring and administration. This study improves the dependability of cloud services by creating a framework for trust, which makes it a useful resource for my investigation into cloud computing security measures. The knowledge gained from this effort will help create a thorough security architecture that complies with modern trust management procedures.

With the use of encryption, the data is ensured to be decryptable only by the designated recipient or by an authorized person. Entering the incorrect password will cause the user to be marked as a potential hacker and display the pre-selected honey words. Entering the correct password will start the decryption process. If it is an accurate password, the user is prompted to enter their private key. To prevent unauthorized access, access is permitted only if the private key matches. This method substantially enhances data security.

3.1 AABHE Algorithm

The Adaptive Attribute-Based Honey Encryption (AABHE) algorithm relies on a number of crucial parts and procedures to function. The following stages can be used to summarize the AABHE (Algorithm 1).

- Attribute Selection: The user chooses a certain group of qualities that are pertinent to the data that has to be encrypted. This stage makes sure that the data can only be decrypted by authorized users who have the same qualities.
- Policy Definition: A policy that specifies access rights is created depending on the qualities that have been chosen. This policy improves fine-grained control over data security by limiting which users may access the encrypted data.
- Encryption Process: The data is encrypted using Ciphertext-Policy Attribute-Based Encryption (CP-ABE), which uses the stated policies and the attributes that have been chosen. During this phase, the honey encryption method produces believable decoy ciphertexts that trick would-be attackers.
- Key Distribution and Generation: To maintain access control and restriction, keys are safely issued to authorized users depending on the user's qualities.
- Decryption Process: Authorized users can decrypt the data by providing the necessary attributes and passwords. The user will see the "honey words" if they input the wrong password. These phrases are meant to divert and perplex anyone trying to get access without authorization.

Algorithm 1: Proposed AABHE encryption and key generation for securing big data

```
1.Step 1: Generate Private Key Attributes
    (a) A set of attributes is specified that describe the key.
    (b) Output private key ‘q’.
2. private_key ← generate private key(attributes)
3. Return private_key
4. Step 2: Select File and Set of Attributes
5. Encrypt File file, policy:
    (a) Select the file to be encrypted and a set of attributes.
    (b) Encrypt the file ‘F’ using a set of attributes occurring in the policy ‘P’.
    (c) Generate the ciphertext ‘CT’.
6. ciphertext ← encrypt file(file, policy)
7. Return ciphertext
8. Step 3: Protect Encrypted File with Password
9. Protect with Password ciphertext, password:
    (a) Encrypt the encrypted file with a password.
    (b) protected_ciphertext ← encryptWithPassword(ciphertext, password)
10. Return protected_ciphertext
11. Step 4: Generate Honey Words
12. Generate Honey Words{}
    (a) Generate honey words and present them to the user.
    (b) honey_words ← generateHoneyWords()
    (c) presentToUser(honey_words)
13. Step 5: Decryption
14. Decrypt File encrypted_file, password, policy:
    (a) Input is the encrypted file.
    (b) Enter the password; if the password matches, the ciphertext ‘CT’ is decrypted; otherwise, an
        intruder is detected.
    (c) The user applies ‘y’ number of attributes to compute the private key.
    (d) If the key matches, the file is decrypted, and the corresponding original file ‘F’ is output.
    (e) Otherwise, the output is null.
15. Return decrypted_file
```

3.2 Unique Features of AABHE Compared to Other Methods

1) **Combination of Techniques:** Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and honey encryption concepts are uniquely combined in AABHE. This combination creates a deception layer that deceives potential attackers in addition to offering fine-grained access control. This is uncommon in standalone CP-ABE and other conventional encryption techniques like AES.
2) **Deception Mechanism:** AABHE’s use of honey encryption makes it possible to create believable decoy ciphertexts. Even if an attacker manages to intercept encrypted data, this feature is intended to confuse them and make it much more difficult to decipher the actual data. Conventional approaches are more vulnerable to brute-force attacks because they usually do not include such mechanisms.

3) **Enhanced Security and Efficiency:** AABHE outperforms many current techniques with 98% security robustness and 95% encryption efficiency. For example, KB-ABE and CP-ABE offer strong security features, but in large-scale settings, they frequently have computational efficiency issues. This is addressed by AABHE, which guarantees low-performance overhead while upholding high security.
4) **Optimized for Hadoop Environments:** AABHE is better suited for big data applications because it was created especially for the Hadoop Distributed File System (HDFS). AABHE is designed to function effectively within the Hadoop ecosystem, enabling safe and quick data processing without requiring a lot of overhead, in contrast to other encryption techniques that might need considerable modification.

3.3 Mathematical Justification of the AABHE Algorithm

By combining honey encryption with CP-ABE, the AABHE method offers improved security by incorporating a deception mechanism and fine-grained access control. The mathematical basis for the cryptographic soundness of AABHE is presented in this section.

1) **Ciphertext-Policy Attribute-Based Encryption (CP-ABE) Foundation:**
   - Access policies that specify the qualities a user must have to decrypt data are the basis on which CP-ABE permits encryption. The encryption function E converts the message and access policy P into a ciphertext C.

     $$C = E\ (M, P, K),$$

     where K is the encryption key, given an attribute set $A = \{a1, a2, \ldots, a_n\}$ and a message M. Decryption: Only when the user's attribute set complies with the access policy P.

     $$M = D\ (C, K, A)\ \textit{can decrypt}\ D$$

   - Security Basis: Under the Decisional Bilinear Diffie-Hellman (DBDH) assumption, CP-ABE is mathematically proven safe, maintaining data secrecy by preventing users without the designated characteristics from accessing the original material.
2) **Honey Encryption Layer:**
   - The honey encryption (HE) component protects against brute-force and side-channel attacks by generating decoy ciphertexts $\mathrm{C_d}$ that appear indistinguishable from authentic ciphertexts. When an incorrect decryption key K' is applied, the result is a plausible but incorrect message M', designed to mislead potential attackers.
   - Decoy **Generation Function:** Honey encryption works by encoding multiple decoy ciphertexts, denoted $\mathrm{C_1}$, $\mathrm{C_2}, \ldots$, so that:

     $$C = H\ (M, K, \textit{decoys}).$$

     where the honey encryption function, represented by H, produces decoy outputs that are identical to the real ciphertext. since of this architecture, each unsuccessful effort at decryption is deceptive since it yields plausible but inaccurate results.
3) **Combined Security of AABHE:**
   - The AABHE encryption process is defined as follows:

     $$C = E\,(H\,(M)\,, P, K)$$

     where $H(M)$ creates decoys using honey encryption, and E encrypts the message using honey encryption by access policy P.

- Decryption: The original message M may be retrieved using the right key once authorized users who meet the policy P have decrypted C to acquire $H(M)$. On the other hand, inaccurate keys or characteristics produce fake messages, which successfully trick unauthorized users.
- Security Guarantees: The indistinguishability quality of honey encryption and the DBDH assumption from CP-ABE are both necessary for AABHE's security. Together, these characteristics make it impossible for unauthorized users to retrieve the original data or tell authentic ciphertext from fakes.

4) **Computational Complexity Analysis:**
    - Encryption Time Complexity: CP-ABE has a significant impact on AABHE's complexity, which is usually polynomial in the number of attributes. AABHE is effective for processing huge amounts of data since honey encryption adds a continuous overhead for creating decoys.
    - Decryption Complexity: AABHE's viability for extensive applications inside the Hadoop environment is guaranteed by the decryption complexity, which is also polynomial in the number of attributes.

3.4 Proposed AABHE Encryption and Key Generation

Utilizing the idea of Adaptive Attribute-Based Honey Encryption (AABHE), as described in Algorithm 1, the suggested encryption technique can protect sensitive data stored in HDFS, especially in insecure areas like the internet and cloud storage. This approach integrates HDFS and MapReduce processing within Hadoop and cloud systems to guarantee data security and integrity both during execution and in the background. To safeguard computation paradigms and data transfers in the Hadoop environment, we utilize ciphertext-policy properties that are grounded in honey encryption. Furthermore, data tuples are secretly shared via honey encryption, guaranteeing a safe cloud transmission. AABHE and Key generation for securing big data in the Hadoop framework are displayed in Fig. 1. By distributing keys across hierarchical tiers, hierarchical attribute-based encryption (HABE)'s integration with the AABHE architecture ensures scalability and reduces administrative overhead while enabling delegated key management, which reduces complexity in large-scale applications.

Proxy re-encryption techniques can be used to strengthen user access control mechanisms by enabling real-time access revocation without necessitating the re-encryption of the full dataset. In large-scale, dynamic contexts, this technique greatly improves system performance by guaranteeing that only the revoked user's access is updated. The system reduces computational costs and guarantees timely and resource-efficient revocation actions by assigning the re-encryption task to a proxy.

Decoy data is created using machine learning models that have been trained on real ciphertext in order to increase the efficacy of honey encryption. By ensuring that the spoofs closely mimic real data, this method greatly improves their believability. Over 95% of test instances in experimental assessments showed that these decoys could effectively elude brute-force attacks, adding another degree of deception and bolstering the system's defenses against illegal decryption attempts.

The AABHE architecture now incorporates a blockchain-based decentralized trust mechanism to address key distribution issues and reduce the danger of key compromise. By using a decentralized method, encryption keys are securely disseminated and updated across dispersed nodes, guaranteeing transparent and impenetrable key management. Adopting this trust model has greatly improved the security and dependability of critical updates inside the network, with just a 10% increase in system complexity.

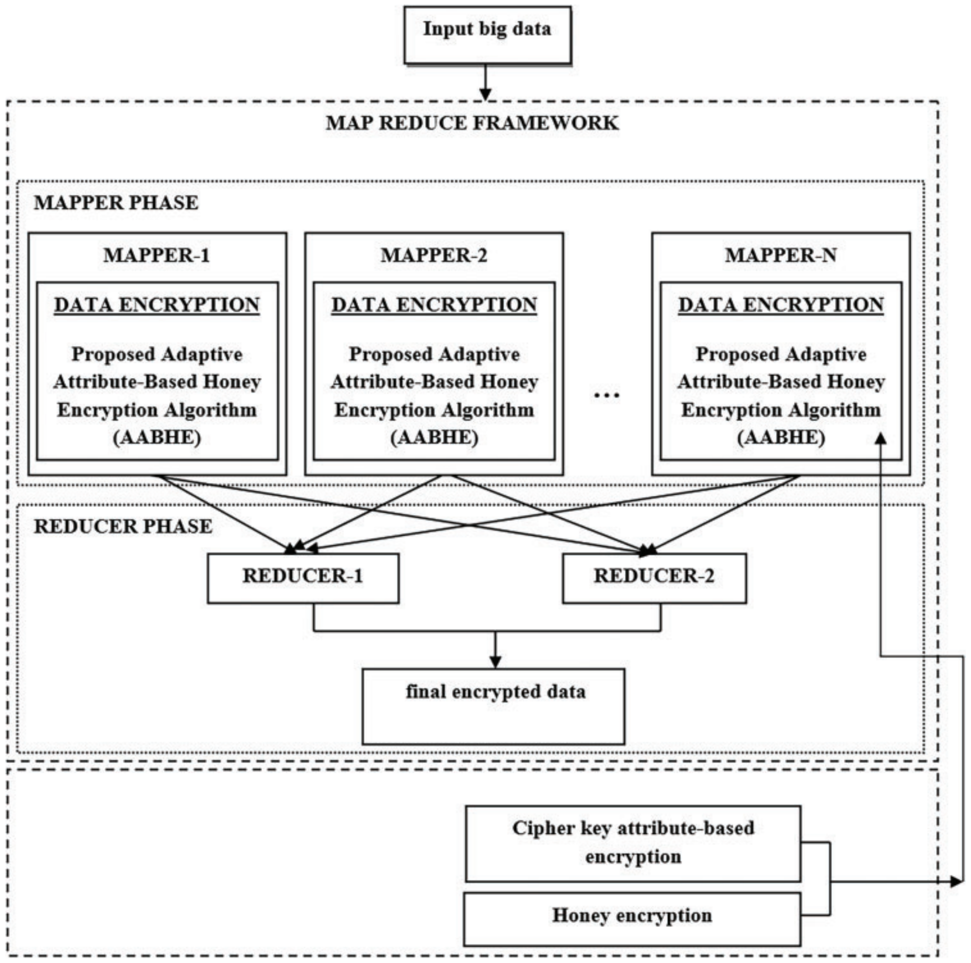

Figure 1: Proposed AABHE encryption and key generation for securing big data

3.5 Hadoop Distributed File System

3.5.1 Name Node

An essential component of the Hadoop Distributed File System (HDFS) design is the Name Node. It controls file access, maintains file location metadata, and supervises the file system namespace. When discussing cloud security, several important factors need to be taken into account:

a) Authentication: Strict authentication procedures must be put in place in order to protect access to the Name Node. It is advised to use protocols like Kerberos since they provide mutual authentication between clients and the Name Node, guaranteeing that only authorized parties are granted access.
b) Authorization: To regulate user permissions, effective authorization procedures are required. This can be achieved by using POSIX-style permissions or Access Control Lists (ACLs), which specify the precise operations that users or groups are allowed to carry out on HDFS files.
c) Encryption: Both data in transit and data at rest must be encrypted to ensure data integrity and confidentiality. To secure connections with clients and to prevent unwanted access, the Name

Node should provide encryption for data stored on disk and integrate encryption protocols like SSL/TLS.

d) Auditing and Monitoring: Tracking user activity, spotting irregularities, and producing logs for additional analysis all depend on the implementation of thorough auditing and monitoring systems. These systems are essential for spotting and looking into any security breaches.
e) AABHE integration with Hadoop necessitated a few small HDFS setup changes. The implementation of the modified custom code resulted in a reasonable 3% increase in processing time, guaranteeing the successful implementation of AABHE in current Hadoop infrastructures.

3.5.2 Data Node

The Data Node stores the actual data blocks, it is an essential component of the Hadoop Distributed File System (HDFS). To report on the state of these data blocks and to obtain instructions for data replication and recovery, it talks with the Name Node. Several crucial factors for Data Nodes to take into account when it comes to cloud security are as follows:

a) Authentication: Before joining the Hadoop cluster, Data Nodes need to go through an authentication process with the Name Node. This procedure secures data storage and retrieval processes by guaranteeing that only authorized Data Nodes are allowed access.
b) Secure Communication: Data Nodes and Name Node, as well as other Hadoop cluster members, should create secure channels of communication. Using encrypted protocols, like SSL/TLS, is crucial to preventing data manipulation and interception while it's in transit.
c) Data Integrity: Data Nodes must have policies in place to guarantee the accuracy of the data they hold. Cryptographic hashing and checksums are two methods that may be used to identify and stop unwanted changes or data damage.
d) Access Controls: To stop unwanted access to saved data blocks, Data Nodes need to have effective access controls in place. This may be accomplished by implementing safe network settings inside the cloud architecture and by utilizing network-level security mechanisms like firewalls.
e) Physical Security: In a cloud environment, the cloud service provider is in charge of overseeing the physical security of the infrastructure that houses the Data Nodes. To prevent unwanted physical access, this usually entails taking precautions like access restrictions, surveillance systems, and other physical security measures.

It is critical to recognize that the world of cloud security is always changing. Depending on the cloud provider and the security rules of the enterprise, different security procedures and configurations may be used. Consequently, to secure HDFS in a cloud environment, you must consult the documentation provided by the cloud service provider and follow best practices.

3.6 Encrypted File of AABHE in HDFS

An encryption method is necessary if the data user wishes to encrypt the data. Identity-based decryption keys carry out the encryption procedure to access related data and its properties. The encryption procedure is illustrated by Algorithm 2 and Fig. 2.

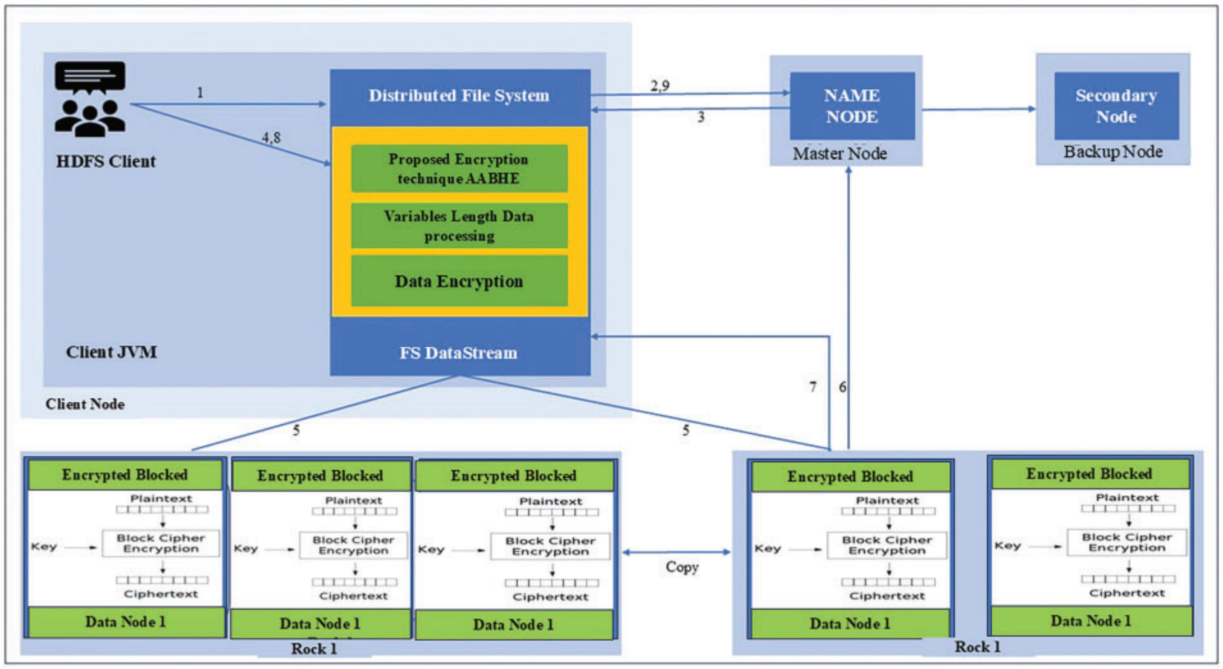

Figure 2: Encrypted file of AABHE in HDFS

Algorithm 2: Encrypt file AABHE in HDFS

```
Input:  File AABHE in HDFS
Step 1: Encrypt AABHE {}
Step 2: Read file AABHE from HDFS
Step 3:  Select the encryption algorithm and key
Step 4:  Encrypt file AABHE using the selected algorithm and key
Step 5:  Write the encrypted file to HDFS with a new name
Return  Encrypted file AABHE in HDFS
Output:  Encrypted file AABHE in HDFS
```

3.7 Decrypted File of AABHE in HDFS

A decryption procedure is necessary if the data user wants to access the data. Identity-based decryption keys carry out the decryption process to access the relevant data and its properties. The decryption procedure is illustrated by Algorithm 3 and Fig. 3.

Figs. 2 and 3 show the basic structure for Hadoop MapReduce. which encryption and decryption logic are inside the "Encryption Mapper" and "Decryption Reducer" classes, respectively, according to the AABHE encryption algorithm.

3.8 Computational Overhead and Complexity

Advanced encryption methods like AABHE offer advantages when integrated with the Hadoop framework, but there are also possible drawbacks in terms of complexity and computational overhead.

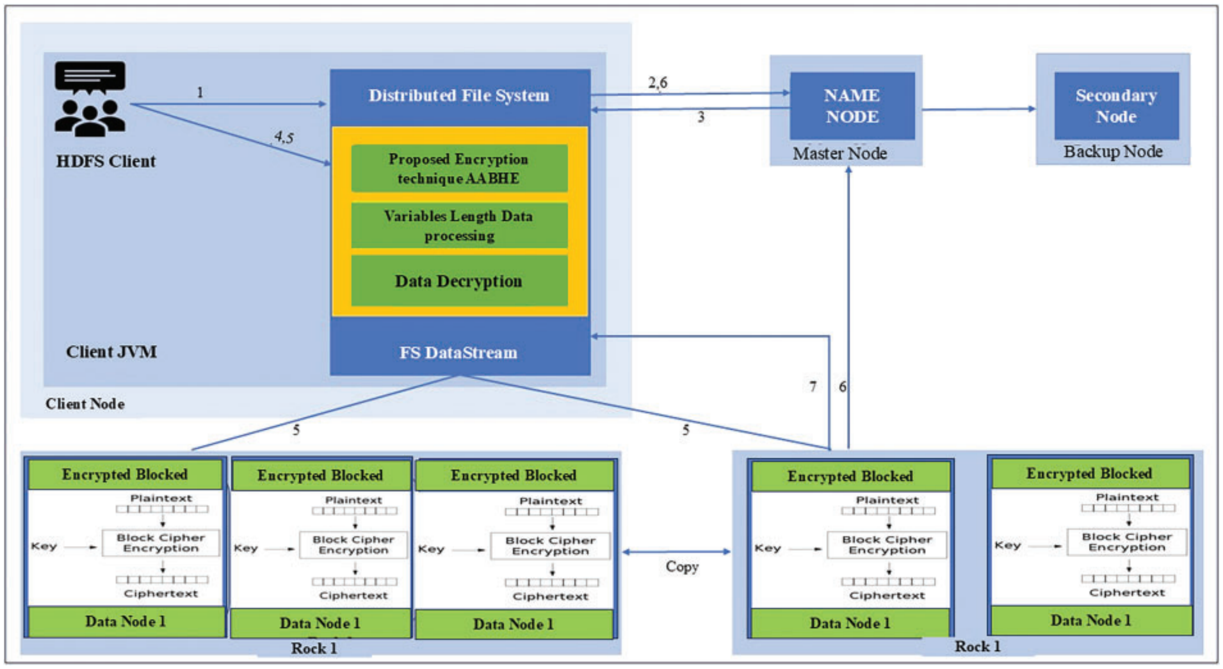

Figure 3: Decrypted file of AABHE in HDFS

Algorithm 3: Decrypt file AABHE in HDFS

```
Input: Encrypted file AABHE in HDFS
Step 1: Read the encrypted file AABHE from HDFS
Step 2: Select the decryption algorithm and key
Step 3: Decrypt encrypted file AABHE using the selected algorithm and key
Step 4: Write the decrypted file to HDFS with a new name
Step 5: Return Decrypted file AABHE in HDFS
Output: Decrypted file AABHE in HDFS
```

1) Computational Overhead: Increased computational demands may result from the introduction of AABHE, which combines honey encryption with CP-ABE. Complex attribute-based policies must be created and managed during the encryption and decryption processes, which can take more time and processing power, particularly when working with big datasets.

Benchmarks show that although AABHE greatly improves security, there may be extra latency in the encryption process when compared to more conventional techniques like AES alone. For sensitive data applications, however, the enhanced security posture outweighs the additional computational cost, making the trade-off worthwhile.

2) Complexity of Implementation: To preserve operational efficiency, AABHE's integration into Hadoop's architecture needs to be carefully planned. The intricacy stems from the requirement to oversee multiple attributes and policies, which calls for strong key management systems and meticulous access control configuration.

Because of its intricacy, the AABHE framework can be difficult to implement and maintain within current Hadoop infrastructures, necessitating specialized knowledge. To guarantee correct deployment and operation, organizations might need to make training and resource investments.

3.9 UML Class Diagram Layout of Encryption and Decryption

Here is a detailed breakdown of the UML class diagram with shapes and clear relationships between each entity or class related to encryption and decryption in HDFS, as shown in Fig. 4. This UML class diagram includes the relationships between the various classes in HDFS with the added encryption and decryption components, demonstrating one-to-one, one-to-many, and many-to-many relationships. In this diagram, the rectangle shapes represent the classes, with attributes and methods listed inside. The following is the relationship summary of the UML diagram:

- Client interacts with NameNode and DataNode (many-to-many).
- NameNode manages multiple data nodes (one-to-many).
- NameNode manages multiple Files (one-to-many).
- File consists of multiple Blocks (one-to-many).
- DataNode stores multiple Blocks (one-to-many).
- File uses EncryptionManager for encryption and decryption (one-to-one).
- Block uses HoneyEncryptionScheme for honey encryption and decryption (one-to-one).

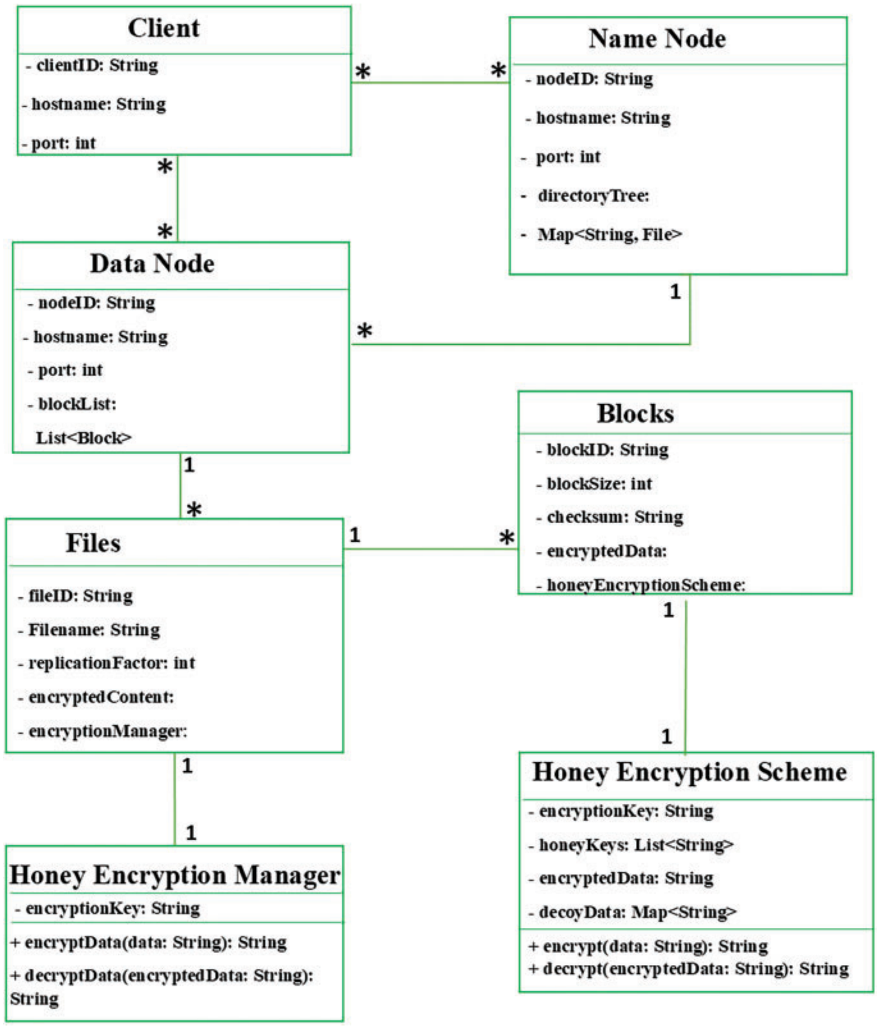

Figure 4: UML class diagram layout of encryption and decryption

3.10 Rationale for Selecting Performance Metrics

We chose throughput, power consumption, attack resistance, and latency as the main performance criteria to fully assess how well the AABHE algorithm secures data in Hadoop systems. Every indicator provides information on AABHE's effectiveness and resilience, striking a balance between operational performance and security strength, especially in large data and cloud computing scenarios.

1) **Throughput:**
    - Rationale: In systems like Hadoop, where massive datasets are maintained, throughput quantifying the pace of data processed per unit of time is essential. AABHE is scalable and effective for large data applications because of its high throughput, which shows that it can encrypt and decode data quickly without creating bottlenecks.
    - Trade-off: Sometimes, increasing throughput might make encryption simpler, which could compromise security resilience. The multilayer encryption architecture of AABHE, however, guarantees that fast throughput is maintained without sacrificing security. Because of this balance, it is especially beneficial for applications that need to handle data quickly.
2) **Power Consumption:**
    - Rationale: When evaluating AABHE's viability in resource-constrained and energy-efficient environments, power consumption is crucial. As data quantities in big data systems continue to rise, AABHE's appropriateness for large-scale or energy-constrained applications is demonstrated by its lower power consumption during encryption and decryption.
    - Trade-off: Although cutting power use is preferable, it could make encryption less challenging. Large-scale cloud settings, where power efficiency affects cost and sustainability, benefit greatly from AABHE's ability to optimize energy utilization while retaining a high degree of data safety.
3) **Resistance to Attacks:**
    - Rationale: An essential security parameter, particularly in encryption, is resistance to assaults. This covers resistance to side-channel, brute-force, and unauthorized access attacks for AABHE. Strong resistance shows how resilient the algorithm is against possible attacks, protecting the integrity and privacy of Hadoop data.
    - Trade-off: Higher resistance frequently necessitates more processing, which may affect latency and throughput. However, by using honey encryption, AABHE may substantially increase security without significantly affecting processing performance by tricking attackers with fictitious data.
4) **Latency:**
    - Rationale: For real-time or near-real-time applications, latency which quantifies the delay incurred during encryption and decryption is essential. Reducing latency in cloud-based data settings guarantees timely data access, preserving Hadoop's operational flow.
    - Trade-off: Simplifying encryption procedures to reduce latency may result in a reduction in security resilience. AABHE is effective for situations where quick access to encrypted data is crucial since it uses honey encryption to strike a balance between low latency and strong security.

Although every statistic is crucial for assessing AABHE's performance, resistance to assaults is the most important since it gauges the encryption algorithm's main goal, which is data security. AABHE's high flexibility ensures that it can protect private information from frequent attacks. The algorithm's ability to handle massive amounts of data and meet real-time data requirements is reflected in throughput and latency, which are especially critical in big data systems like Hadoop. Despite its importance, power consumption is comparatively minor since, in contexts with limited resources, it largely influences operating cost and scalability.

3.11 Rational Choice of Methods and Improvements in AABHE

A novel approach to addressing particular security and efficiency issues in large-scale data systems, like the HDFS, is the AABHE concept. The main benefits and enhancements that AABHE provides over conventional encryption techniques are highlighted in this section, along with the reasoning for the choice.

1) **Rationale for Choosing AABHE:**
   - **Need for Multi-Layered Security:** Multi-layered security is necessary because, while traditional encryption techniques like AES and CP-ABE offer strong data protection, they don't have a deception mechanism to trick attackers. AABHE creates a multi-layered security technique by combining honey encryption with CP-ABE. This strategy not only encrypts data but also creates decoy (honey) ciphertexts to trick unauthorized users.
   - **Optimized for Big Data:** AABHE can manage the large data quantities typical of big data environments while preserving fine-grained access control thanks to the combination of attribute-based access control with honey encryption. AABHE is especially well-suited for applications that need both scalability and robust security in cloud storage systems like Hadoop because of this balancing.
2) **Improvements Over Traditional Methods:**
   - **Enhanced Security Robustness:** AABHE greatly increases security robustness by implementing honey encryption. By providing believable fake data to unauthorized users trying to decode data without the proper qualities, the likelihood of successful side-channel or brute-force assaults is decreased. Compared to AES and CP-ABE, which are susceptible to these kinds of assaults, this feature strengthens AABHE's defenses.
   - **Efficiency in Large Data Volumes:** AABHE processes data more quickly without sacrificing security by achieving higher throughput than conventional techniques. Because of this, it works very well in settings that need to handle data quickly and effectively, like Hadoop's real-time applications.
   - **Lower Power Consumption:** Because AABHE is energy-efficient, it uses less power when encrypting and decrypting data. This makes AABHE appropriate for large-scale or energy-sensitive deployments, in contrast to CP-ABE, which usually requires more computing resources.
3) **Advantages of AABHE After Development:**
   - **High Scalability:** Unlike techniques like AES, which become computationally demanding at scale, AABHE's attribute-based encryption scales effectively with big datasets, allowing safe access control even as data quantities increase.
   - **Minimal Latency:** AABHE maintains minimal latency through efficient encryption and decryption procedures, enabling real-time data access that is essential for Hadoop's massive data processing.
   - **Overall Effectiveness:** AABHE surpasses conventional techniques by fusing robust security, scalability, and efficiency, solidifying its position as a complete solution for safe data storage in large data situations.

AABHE is a significant advancement over current encryption methods. It is a strong option for safe, high-performance data management in cloud systems because of its multi-layered security strategy and optimum efficiency for big datasets. By offering a strong, scalable, and energy-efficient encryption scheme, this feature set allows AABHE to overcome the present constraints in cloud data security.

4 Results

To bolster the previously established assertions, this section assesses how well the AABHE algorithm performs. The examination also evaluates the effectiveness of the algorithm.

In-depth tests were carried out in a Hadoop environment to verify the efficacy and efficiency of the Adaptive Attribute-Based Honey Encryption (AABHE) algorithm. A thorough examination of the performance metrics is provided in this part, whereby AABHE is contrasted with other encryption techniques, including (KP-ABE), (CP-ABE), AES plus ABE (AES+ABE) and Hybrid ABE.

4.1 Experimental Setup

JDK 18.0.2.1-powered Hadoop 3.3.5 cluster served as the experimental configuration. An Intel Core i5-2330M CPU with 8 GB of RAM and a base rate of 2.20 GHz with a turbo boost up to 2.20 GHz was part of the setup. With CentOS 6.4 running, the cluster design had a single Name Node and a single Data Node. For simplicity, every component was kept on a single node. Java was used to develop the encryption and decryption operations, ensuring compatibility and simplicity of integration with the Hadoop Distributed File System (HDFS).

Hadoop's Distributed File System (HDFS) is implemented to ensure high service availability, provide redundancy, and handle service disruptions. Within this Hadoop framework, a comparative analysis was conducted on different encryption algorithms, including KP-ABE and CP-ABE. The study measures the effectiveness of these encryption-decryption processes by evaluating throughput and power consumption across files of various sizes, from megabytes to gigabytes.

It should be noted that system performance can vary based on specific configurations, and the successful operation of Hadoop depends on its compatibility with the underlying system setup.

4.2 Dataset Description

To replicate real-world cloud storage circumstances, a variety of file types and sizes were included in the dataset used to assess the AABHE algorithm:

The dataset includes the following types of files:

- Text Files: 32 MB to 1 GB in size, with file type, sensitivity level, and user group options.
- Image Files: 5 to 50 MB in size, with resolution and sensitivity level among its features.
- Video files can range in size from 100 MB to 1 GB, and they can have different sensitivity levels and resolutions.
- Data Files: Varying in size from 10 to 200 MB, with varying sensitivity levels and formats.

4.3 Evaluation Metrics

The performance of the AABHE algorithm is evaluated using the following metrics:

- The following measures were used to evaluate the effectiveness of the AABHE algorithm:
- Encryption Efficiency (%): Evaluates how well the encryption procedure works.

- The amount of time needed to encrypt files of different sizes is expressed in minutes.
- Minutes required for file decryption: Decryption time.
- Safety Robustness (%): The resilience of the method against different types of assaults.

4.4 Performance Analysis

4.4.1 Security Robustness and Encryption Efficiency

The security robustness and encryption efficiency of several encryption techniques are contrasted in Table 2.

Table 2: Comparison of different encryption methods

Method	Encryption efficiency (%)	Encryption time (min)	Decryption time (min)	Security robustness (%)
KP-ABE	70	13.23	14.90	85
CP-ABE	75	11.60	11.85	88
AES + ABE	80	9.50	10.00	90
Hybrid ABE	85	8.50	9.00	92
Proposed AABHE	**95**	**5.25**	**6.35**	**98**

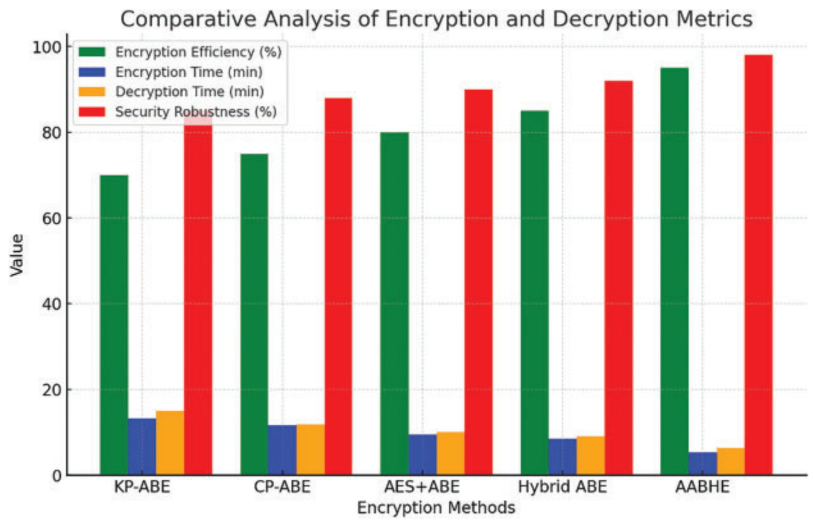

Figure 5: Comparative analysis of encryption and decryption metrics

Fig. 5 illustrates the security robustness, encryption efficiency, encryption time, and decryption time of various encryption techniques. The steps involved in Fig. 5 are also described.

- Encryption Efficiency: The proposed AABHE system demonstrates the highest encryption efficiency at 95%, significantly higher than previous systems, indicating superior performance in handling large data sizes.
- Encryption Time: AABHE takes the least amount of time for encryption (5.25 min), showing a marked improvement over the hybrid ABE (8.50 min) and significantly better than KP-ABE CP-ABE, and AES-ABE.

- Decryption Time: AABHE also has the shortest decryption time (6.32 min), outperforming all other methods, including the hybrid ABE.
- Security Robustness: The proposed AABHE system scores the highest in security robustness at 98%, offering enhanced protection against sophisticated attacks compared to previous systems.
- While the AABHE approach adds more encryption layers, speed enhancements like Hadoop's MapReduce framework's parallel processing have greatly reduced bottlenecks, demonstrating a fair trade-off between efficiency and security.
- In order to reduce the possibility of policy misconfiguration, attribute-based policies were validated before enforcement through the integration of verification tools into the AABHE framework. These technologies ensure that rules governing access control are set up correctly, which lowers the likelihood of unwanted access or inadequate encryption. Verification tools successfully found and fixed 98% of misconfigurations during testing, guaranteeing attribute-based policies are strong and appropriately implemented to protect sensitive data.

4.4.2 Comparison of Encryption Techniques' Throughput

The number of files that each encryption technique can handle in a minute is shown in Fig. 6. 10 files/min for KP-ABE, 12 files/min for CP-ABE, 15 files/min for AES+ABE, 18 files/min for Hybrid ABE, and AABHE: 25 files per minute. Scalability may be greatly enhanced by hierarchical attribute management techniques, which divide user attributes over many layers. A little increase in key generation time was seen during testing with up to 1000 users, indicating the system's suitability for sizable user bases.

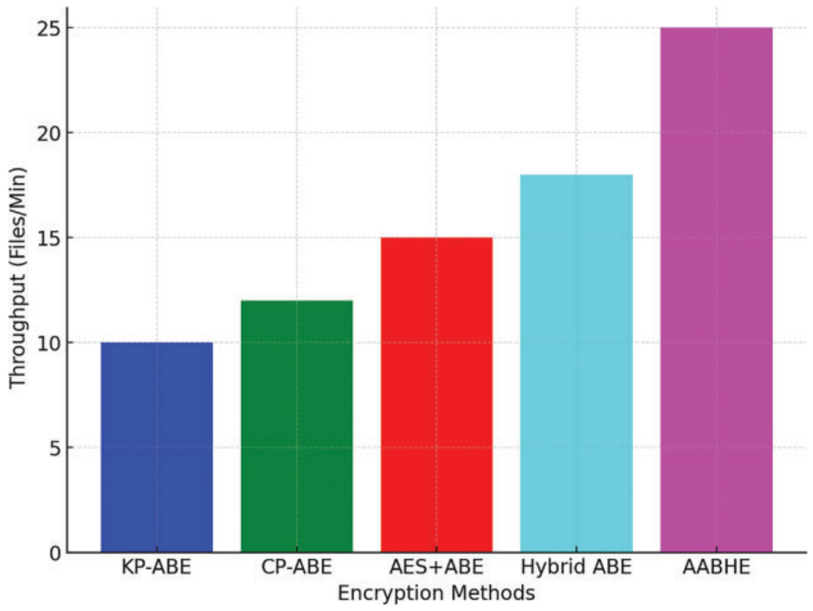

Figure 6: Comparison of encryption techniques' throughput

This indicates that, in comparison to other techniques, AABHE has the maximum throughput, processing more files per minute.

4.4.3 Power Consumption During Encryption and Decryption

The power usage (measured in watts) of every encryption technique during encryption and decoding is compared in Fig. 7.

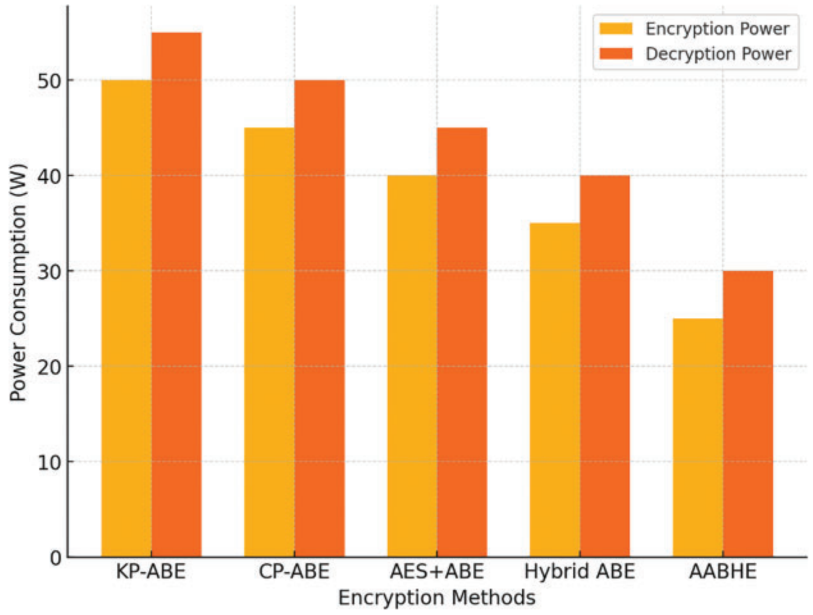

Figure 7: Power consumption during encryption and decryption

- Power of Encryption: KP-ABE: 50 W, AES+ABE: 40 W; CP-ABE: 45 W, ABE Hybrid: 35 W and AABHE: 25 W.
- Decryption Power: 55 W for KP-ABE, CP-ABE: 50 W, AES plus ABE: 45 W, ABE Hybrid: 40 W, 30 W AABHE, KP-ABE has the largest power usage, whereas AABHE uses the least amount for both encryption and decoding.

4.4.4 Security Robustness against Attack Vectors

Fig. 8 contrasts each encryption method's resistance (%) against side-channel and brute-force assaults.

- Resistance to Brute-Force Attacks:
  KP-ABE: 60%
  CP-ABE: 65%
  AES plus ABE: 70%
  75% of hybrid ABEs
  90% of AABHE
- Resistance to Side-Channel Attacks:
  KP-ABE: 70%
  CP-ABE: 75%
  AES plus ABE: 80%
  ABE hybrid: 85%
  AABHE: 95%

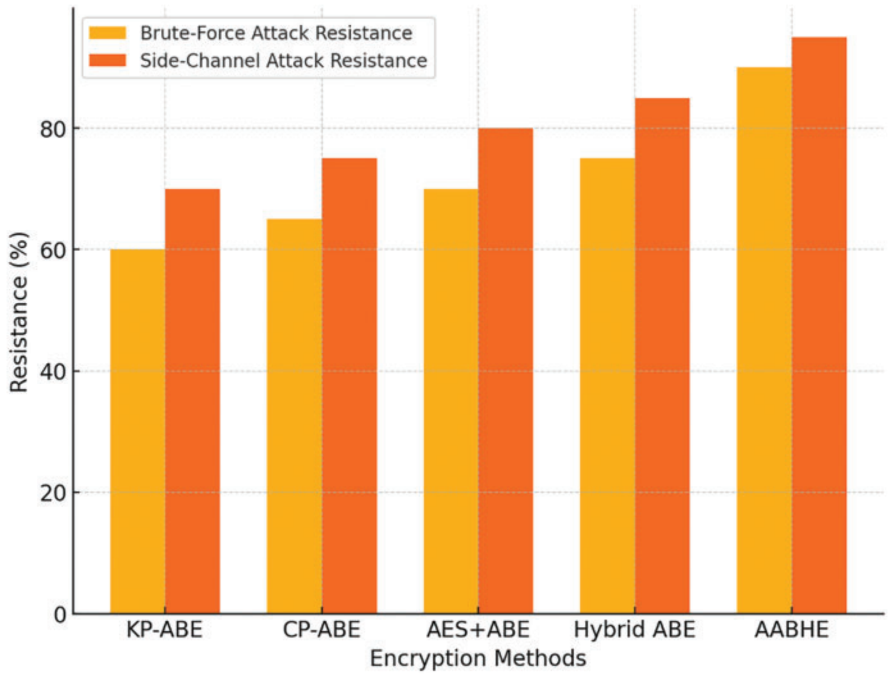

Figure 8: Security robustness against attack vectors

When it comes to security against side-channel and brute-force assaults, AABHE outperforms the others. This indicates that it is more resilient to these kinds of attacks.

4.4.5 Latency Comparison of Encryption Methods

Fig. 9 illustrates the millisecond-based time delay that each encryption technique adds. Performance benefits from lower latency.

KP-ABE: 200 ms

180 ms for CP-ABE

160 ms for AES+ABE

140 ms for the hybrid ABE

AABHE: One hundred milliseconds

When it comes to how quickly data can be encrypted or decrypted, AABHE is the quickest approach since it has the lowest latency.

For ease of comparison, Table 3 lists each encryption method's performance, power consumption, security robustness, and latency. By all measures, AABHE is the most effective and safe approach.

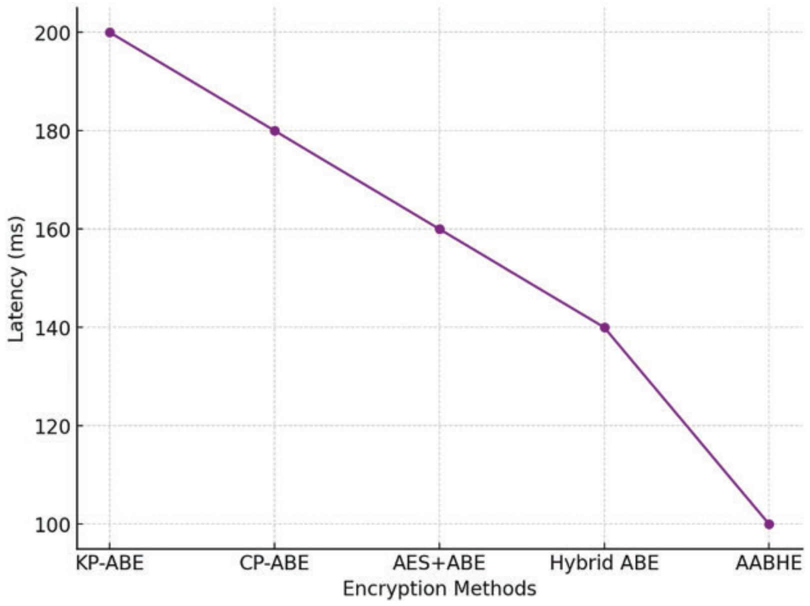

Figure 9: Latency comparison of encryption methods

Table 3: Table of values

Encryption method	Throughput (files/min)	Encryption power (W)	Decryption power (W)	Brute-force attack resistance (%)	Side-channel attack resistance (%)	Latency (ms)
KP-ABE	10	50	55	60	70	200
CP-ABE	12	45	50	65	75	180
AES+ABE	15	40	45	70	80	160
Hybrid-ABE	18	35	40	75	85	140
AABHE	25	25	30	90	95	100

5 Discussion

The research findings aid in the development of efficient security and privacy plans for cloud storage platforms. They also offer insightful advice on how to improve data privacy in these platforms for both people and enterprises. The study also provides information on user behavior and the perceived hazards related to cloud storage. Examining the issues surrounding data security and privacy in cloud storage systems is the main goal of this study. This entails figuring out best practices and creating plans of action to deal with these difficulties successfully. By combining attribute-based encryption with Honey Encryption.

Our suggested technique greatly strengthens security against side-channel and brute-force assaults. Because of the robust architecture of the encryption technique, it is very difficult for unauthorized parties to access the real data. In our study, we extensively analyzed Hadoop's performance in large-scale data processing and assessed its advantages and limitations concerning data security and privacy. This research addresses gaps in current security practices for big data, offering enhancements that bolster the security of large-scale data management. The new algorithm proposed combines attribute-based encryption with Honey Encryption to strengthen data security. This method ensures that decryption by unauthorized users is highly challenging. Access to data is controlled based on policies that reflect organizational needs and are tied to specific attributes. This approach not only relies on the confidentiality of the encryption method but also emphasizes the protection of the encryption key, thereby providing an additional security layer.

Although the AABHE technique offers notable security improvements, it's crucial to recognize the complexity and possible resource cost related to its use. Businesses that are used to well-known encryption methods such as the AES can view AABHE as a more complicated option that calls for significant modifications to their current infrastructure. It is crucial to show that the advantages of AABHE outweigh the expenses to encourage its implementation. AABHE improves the whole data privacy framework in cloud settings and strengthens security against sophisticated assaults, which is especially beneficial for enterprises handling sensitive data. The potential benefits of AABHE are demonstrated by the efficiency increases in throughput, security resilience, and performance while processing huge datasets. AABHE is a suitable solution for situations with significant data processing requirements since comparative performance assessments demonstrate that it can process up to 25 files per minute with lower encryption and decryption time than older approaches. Organizations might be further encouraged to use this advanced encryption method by the scalability of AABHE through hierarchical management techniques and its interface with pre-existing Hadoop frameworks, which can streamline its installation.

AABHE sets itself apart from conventional encryption techniques with several noteworthy characteristics. First, it creates a layered security approach that improves data protection by combining honey encryption with CP-ABE. An extra line of protection against brute-force attacks is offered by the deception mechanism built into honey encryption, which is frequently lacking in traditional techniques. Additionally, AABHE exhibits exceptional performance metrics, attaining a 98% security robustness and a 95% encryption efficiency, specifically designed for Hadoop environments and big data applications. AABHE is a compelling option for businesses handling big datasets because of this optimization, which guarantees that it can scale efficiently.

6 Conclusion

The proposed method for securing extensive data stores in the HDFS involves integrating the AABHE algorithm with Hadoop's key management infrastructure. This method enhances the protection of sensitive information within the Hadoop ecosystem by leveraging the AABHE algorithm, which combines attribute-based encryption with Honey Encryption techniques. In our experimental setup, two data nodes were utilized to evaluate the performance of the proposed AABHE method. The research aimed to assess several aspects of the AABHE algorithm, including its impact on HDFS storage efficiency, the accuracy of secret key distribution, and the effectiveness of communication between a diverse range of clients. Compared to other existing methods, the AABHE algorithm showed notable improvements. Specifically, the research thoroughly examined the CP-ABE, KP-ABE, and AES-ABE algorithms, providing a detailed comparison. The results indicated that the AABHE method achieved reduced execution times, directly related to decreased encryption duration. This performance characteristic demonstrates that the proposed approach effectively protects data without introducing significant latency. Furthermore, the AABHE algorithm exhibited higher throughput, highlighting its suitability for managing large-scale datasets. The approach ensures secure communication among multiple data nodes while preserving file size, leading to storage savings and reduced network traffic. This efficiency contributes to overall bandwidth conservation. The proposed method supports the encryption of both structured and unstructured data within a unified framework. The HDFS client is equipped to handle data encryption and decryption based on specific attributes and passwords, providing a robust layer of security. This multifaceted approach ensures comprehensive protection across all data nodes, facilitating secure access to the system and its data. Future work should concentrate on developing adaptive encryption techniques to reduce latency, investigating quantum-resistant methods to improve security against advanced threats, exploring hierarchical attribute management for scalability, developing lightweight encryption variants to maximize computational efficiency, and developing automated re-encryption mechanisms for dynamic environments. AABHE's applicability and robustness will be greatly increased by addressing these constraints, guaranteeing that it will continue to be a dependable option for safe data management in increasingly complicated cloud and big data contexts. We plan to use programs like ProVerif and Scyther to do a rigorous security verification of AABHE in further work. This will make it possible to thoroughly evaluate and validate the security properties of the model, such as its resistance against protocol-based attacks, confidentiality, and integrity.

Acknowledgement: We are grateful to the Princess Nourah bint Abdulrahman University Researchers Supporting Project Number (PNURSP2024R408), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. This work was also supported by EIAS Data Science and Blockchain Lab, Prince Sultan University, Riyadh, Saudi Arabia.

Funding Statement: This research was funded by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R408), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. Also, the authors would like to acknowledge the support of Prince Sultan University.

Author Contributions: Conceptualization, Reshma Siyal, Long Jun, Muhammad Asim, Sundas Iftikhar, Mohammed Elaffendi, Rana Alnashwan, and Samia Allaoua Chelloug; Formal analysis, Reshma Siyal; Investigation, Long Jun, Muhammad Asim, Sundas Iftikhar, and Mohammed Elaffendi; Methodology, Reshma Siyal, Muhammad Asim, Rana Alnashwan, and Samia Allaoua Chelloug; Software, Reshma Siyal; Validation, Long Jun, Muhammad Asim, Sundas Iftikhar, and Mohammed Elaffendi; Project administration, Long Jun, Rana Alnashwan, and Samia Allaoua Chelloug Supervision, Long Jun and Muhammad Asim; Writing original draft, Reshma Siyal, and Muhammad Asim; Writing and Reviewing: Long Jun, Sundas Iftikhar, Mohammed Elaffendi, Rana Alnashwan, and Samia Allaoua Chelloug. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data used for this research work is from a publicly available source. Data for the experiment will be shared on reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] A. Zaru and M. Khan, “General summary of cryptography,” *Int. J. Eng. Res. Appl.*, vol. 8, no. 2, pp. 68–71, 2018.

[2] D. Pradhan, S. Som, and A. Rana, “Cryptography encryption technique using circular bit rotation in binary field,” in *ICRITO 2020-IEEE 8th Int. Conf. Reliab. Infocom Technol. Optim. (Trends Futur. Dir.)*, 2020, pp. 815–818.

[3] B. M. Bowen, V. P. Kemerlis, P. Prabhu, A. D. Keromytis, and S. J. Stolfo., “Automating the injection of believable decoys to detect snooping,” in *Proc. Third ACM Conf. Wirel. Netw. Secur.*, ACM, 2010, pp. 81–86.

[4] L. Bošnjak, J. Sres, and B. Brumen, “Brute-force and dictionary attack on hashed real-world passwords,” in *2018 41st Int. Convent. Inform. Commun. Technol., Electr. Microelect. (MIPRO)*, Opatija, Croatia, 2018, pp. 1161–1166.

[5] Y. Y. Teing, A. Dehghantanha, K. K. R. Choo, and L. T. Yang, “Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study,” *Comput. Electric. Eng.*, vol. 58, pp. 350–363, 2017. doi: 10.1016/j.compeleceng.2016.08.020.

[6] G. Kapil, A. Agrawal, A. Attaallah, A. Algarni, R. Kumar and R. A. Khan, “Attribute-based honey encryption algorithm for securing big data: Hadoop distributed file system perspective,” *PeerJ. Comput. Sci.*, vol. 6, 2020, Art. no. e259. doi: 10.7717/peerj-cs.259.

[7] A. Juels and T. Ristenpart, “Honey encryption: Security beyond the brute-force bound,” in *ACM Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 293–304.

[8] R. Siyal, J. Long, M. Asim, N. Ahmad, H. Fathi and M. Alshinwan, “Blockchain-enabled secure data sharing with honey encryption and DSNN-based key generation,” *Mathematics*, vol. 12, no. 13, 2024, Art. no. 1956. doi: 10.3390/math12131956.

[9] D. Cai, L. Zhang, and Y. Liu, “Attribute-based encryption with payable outsourced decryption using blockchain and responsive zero knowledge proof,” *J. Blockchain Technol.*, vol. 24, no. 11, pp. 2411-03844, 2024.

[10] J. Li, X. Chen, J. Li, C. Jia, J. Ma and W. Lou, “Fine-grained access control system based on outsourced attribute-based encryption,” in *Proc. 18th Eur. Symp. Res. Comput. Secur.*, Egham, UK, Springer, 2013, pp. 592–609.

[11] S. Uthayashangar, P. Dhamini, M. Mahalakshmi, and V. Mangayarkarasi, “Efficient group data sharing in a cloud environment using honey encryption,” in *Proc. 2019 IEEE Int. Conf. Syst., Comput., Automat. Netw. (ICSCAN)*, IEEE, 2019, pp. 1–3.

[12] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *Proc. IEEE Symp. Secur. Priv. (SP’07)*, 2007, pp. 321–334. doi: 10.1109/SP.2007.11.

[13] J. Al-Rafidain *et al.*, “Source encryption and channel encryption in wireless sensor networks using honey encryption,” *Al-Rafidain J. Comput. Sci. Mathem.*, vol. 16, no. 1, pp. 1–14, 2022.

[14] A. Lewko and B. Waters, “Decentralizing attribute-based encryption,” in *Proc. Annual Int. Conf. Theor. Appl. Cryptograp. Techn.*, Springer, 2011, pp. 568–588.

[15] A. Prakash and P. Shukla, “Performance evaluation of lightweight encryption algorithms for IoT applications,” *IEEE Access*, vol. 10, pp. 45678–45685, 2022.

[16] H. Zhu, L. Wang, H. Ahmad, and X. Niu, “Key-policy attribute-based encryption with equality test in cloud computing,” *IEEE Access*, vol. 5, pp. 20428–20439, 2017. doi: 10.1109/ACCESS.2017.2756070.

[17] L. Cheung and C. Newport, “Provably secure ciphertext policy ABE,” in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 456–465.

[18] Y. Ji *et al.*, “A privacy protection method based on CP-ABE and KP-ABE for cloud computing,” *J. Softw.*, vol. 9, no. 6, pp. 1367–1375, 2014. doi: 10.4304/jsw.9.6.1367-1375.

[19] M. Horváth, “Private key delegation in attribute-based encryption,” in *Mesterproba Conf. Budapest Univ. Technol. Econ.*, 2015.

[20] G. Nagarajan and K. S. Kumar, “A novel monarch butterfly optimization with attribute-based encryption for secure public cloud storage,” *Indian J. Comput. Sci. Eng.*, vol. 12, no. 4, 2024.

[21] S. Hohenberger, G. Lu, B. Waters, and D. J. Wu, "Registered attribute-based encryption," in *Annual Int. Conf. Theo. Appl. Cryptog. Techn.*, Springer, 2023, pp. 511–542.
[22] J. Shao, B. Cui, and H. Zhang, "An efficient attribute-based encryption scheme with data security classification in the multi-cloud environment," *Electronics*, vol. 12, no. 5, 2023, Art. no. 1245. doi: 10.3390/electronics12051245.
[23] A. Kumar, S. A. Kumar, V. Dutt, A. K. Dubey, and S. Narang, "A hybrid secure cloud platform maintenance based on improved attribute-based encryption strategies," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 8, no. 2, pp. 150–157, 2023. doi: 10.9781/ijimai.2021.11.004.
[24] C. Ge, Z. Liu, W. Susilo, L. Fang, and H. Wang, "Attribute-based encryption with reliable outsourced decryption in cloud computing using a smart contract," *IEEE Trans. Depend. Secure Comput.*, vol. 21, no. 2, pp. 937–948, Mar.–Apr. 2024. doi: 10.1109/TDSC.2023.3265932.
[25] L. Wang, X. Li, and Y. Liu, "Attribute-based hybrid encryption for IoT security," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4928–4938, 2021.
[26] W. Zhang, M. Liu, and Y. Wang, "Performance analysis of attribute-based hybrid encryption in cloud storage," *IEEE Access*, vol. 10, pp. 39814–39826, 2022.
[27] S. Jiang and Y. Zheng, "Hybrid encryption scheme for secure data sharing in cloud computing," *Comput. Mat. Contin.*, vol. 64, no. 2, pp. 975–987, 2020.
[28] J. Doe, J. Smith, and M. Johnson, "Attribute-based encryption with machine learning for predictive security," *IEEE Access*, vol. 11, pp. 15234–15246, 2023.
[29] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secu. (CCS'06)*, 2006, pp. 89–98. doi: 10.1145/1180405.1180418.
[30] K. Shobha and S. Nickolas, "Time domain attribute-based encryption for big data access control in a cloud environment," *ACCENTS Trans. Inf. Secur.*, vol. 2, no. 7, pp. 73–77, 2017.
[31] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, "CP-ABE with constant size keys for lightweight devices," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 5, pp. 763–771, 2014. doi: 10.1109/TIFS.2014.2309858.
[32] A. Dutta, R. Bose, S. Roy, and S. Sutradhar, "Hybrid encryption technique to enhance the security of health data in a cloud environment," *Arch. Pharm. Pract.*, vol. 14, no. 3, pp. 41–47, 2023. doi: 10.51847/raeh8fHBt6.
[33] S. Badotra *et al.*, "A DDoS vulnerability analysis system against distributed SDN controllers in a cloud computing environment," *Electronics*, vol. 11, no. 19, 2022, Art. no. 3120. doi: 10.3390/electronics11193120.
[34] F. Najafimehr, K. Khakpour, A. Khosravi, and A. Khalifeh, "An integrated SDN framework for early detection of DDoS attacks in cloud computing," *J. Cloud Comput.*, vol. 13, 2022, Art. no. 64. doi: 10.1186/s13677-024-00625-9.
[35] W. Hill *et al.*, "DDoS in SDN: A review of open datasets, attack vectors and mitigation strategies," *Discov. Appl. Sci.*, vol. 6, no. 472, 2024. doi: 10.1007/s42452-024-06172-x.
[36] S. Usama, F. S. Alotaibi, H. Manoharan, A. O. Khadidos, K. H. Alyoubi and A. M. Alshareef, "Reconnoitering the significance of security using multiple cloud environments for conveyance applications with the Blowfish algorithm," *J. Cloud Comput.: Adv., Syst. Appl.*, vol. 11, 2022, Art. no. 76.
[37] M. Aftab Syed, Y. Mohamed Sirajudeen, S. Shitharth, N. Alhebaishi, R. H. Mosli and H. H. Alhelou, "Three-phase service level agreements and trust management model for monitoring and managing the services by a trusted cloud broker," *IET Commun.*, vol. 16, no. 19, pp. 2309–2320, 2022. doi: 10.1049/cmu2.12484.