SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I4P104

Volume 12 Issue 4, 36-48, April 2025

Original Article

Secure Data Access: Attribute-based Encryption with Key Management Services

Pachipala Yellamma[1], Gandrala Varun[2], Challapalli Charan Kumar[3], Addala Shanmukha Sai Phani Kumar[4], Mahamkali Nagendra Rao[5]

[1,2,3,4,5]*Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India.*

Corresponding Author : pachipala.yamuna@gmail.com

Received: 03 February 2025 Revised: 05 March 2025 Accepted: 06 April 2025 Published: 29 April 2025

Abstract *- The security of cloud computing is enhancing rapidly. Data owners usually encrypt their data so that certain users can decode it, protecting the privacy of data stored in the cloud. If an encrypted device needs to be shared with more individuals than the owner has designated, this could lead to very serious issues. In this research, the proposed end-of-end cloud data encryption utilizing the Attribute-Based Key Management Encryption Algorithm (ABKMEA) produces public and private key pairs for the users participating in the data exchange. The proposed ABKME algorithm is the best approach to address the throughput, performance and less computational time. The proposed algorithm ABKMEA gives the computational time of 128 bits is 1.79 sec, 192 bits is 1.81 sec, 256 bits is 2.02sec, 512 bits is 2.11sec, 1024 bits is 2.42sec,2048 bits is 2.42sec,3072 bits is 2.62sec, and 4096 bits is 2.74sec. Therefore, the proposed Attribute-Based Key Management Encryption Algorithm is the best approach for End to End Encryption associated with encrypting and decrypting the data in cloud computing environments.*

Keywords *- Symmetric and asymmetric keys, Key attributes, Key Management Systems (KMS), Data security, Cloud computing, Data privacy, Key lifecycle management.*

1. Introduction

A new paradigm in computer systems called cloud computing offers customers computing resources, including processing power and data storage, based on their own needs. The benefit of cloud computing is that customers can utilize their computer resources as a service at a low cost from any location at any time over the internet [1]. Data sharing among many users safely is a common case for cloud computing. Since the cloud service provider can no longer be trusted, data secrecy should be guaranteed in this system. Additionally, a suitable revocation mechanism should be offered to manage a user whose credentials have expired.

Additionally, even if a revoked user accesses previously stored data on the cloud server via collusion attacks, the system must demonstrate that the data is unusable and accessible to intermediate malicious users and the cloud server, preserving its confidentiality and integrity. Cryptographic encryption stands as a foundation for guaranteeing the security and protection of information put away in cloud situations. By scrambling their information earlier and outsourcing it to cloud servers, information proprietors build up a defensive boundary that shields touchy data from unauthorized access [2]. This encryption prepares changes over the plaintext information into cipher text, rendering it garbled to anybody missing the comparing unscrambling keys.

This approach is especially vital in open cloud capacity frameworks, where numerous information proprietors may have unmistakable encryption inclinations managed by their special information-sharing prerequisites [3]. At first, an information proprietor might select to share their information with a single client, utilizing encryption methods to produce cipher text available exclusively to that beneficiary. As the information-sharing elements progress, it may be necessary for the information owner to extend and involve more customers. This shift in necessities mandates a modification in the arrangement of the encrypted text to allow decryption by multiple authorized parties [2]. Therefore, cryptographic encryption functions not just as a crucial tool for securing information but also delivers the flexibility needed to adapt to evolving collaboration requirements in cloud-based environments. End-To-End Encryption (E2EE) is a technique for secure communication that prevents unauthorized access to data while it is being transferred from one system or device to another [4].

In an E2EE process, data is encrypted and secured on the sender's device, and only the destination party can decrypt and see the original data. This means that even if the data is captured, third parties will never be able to access or read that data unless they possess the proper encryption key. Below is a breakdown of how E2EE works.

The sender uses a cryptographic technique and encryption key to encrypt the sender's side data, typically produced and handled on the sender's device [8]. Once data becomes encrypted, it continues through an infrastructure that supports secure communication between the devices (such as the internet, a messaging app, or another digital network). Upon arrival at the destination, the data is decrypted using a certain corresponding decryption key and presented back in its original, human-readable form. The keys used for encryption, decryption, cloud storage, etc., remain confidential and not disclosed to a service provider or anyone intermediating the unsafe transmission or channel of data from the sender to the recipient. Once those keys are kept private, even if the transmission channel is compromised or interrupted, the data remains unrevealed protection from bad actors.

End-To-End Encryption (E2EE) serves as a major step for protection. It can be found in messaging apps, email, or file-sharing type services for communication, including valuable or sensitive data or user privacy. E2EE has been determined to be one of the most trusted and secure encryption measures that prevent hackers, spyware, or malicious actors from gaining access to information sent to the recipient to decrypt and not expose it to other intermediaries. E2EE guarantees that only the intended recipient can read and decrypt the shared information from the already encrypted data.

Symmetric encryption, otherwise known as secret key algorithms, is a type of encryption that secures/bad actors away from digital communication and sensitive data, specifically from a method of cryptography. These encryption methods also only depend on a single shared secret key shared by a sender and recipient. In the encryption process, a cryptographic algorithm is applied to a plain text message and is transformed into ciphertext. To anyone who does not possess the appropriate decryption key, the ciphertext appears to be randomized and unreadable. The encrypted ciphertext is then sent over a communication channel - e.g., Internet or private networks - without concern for the method of transmission. After receiving the ciphertext, the recipient applies the same shared secret key to decrypt the ciphertext back into the original plaintext message.

The biggest strength of symmetric encryption is its efficiency and speed, which is ideal for data encryption, secure messaging, financial transactions, and the encryption of cloud storage. However, the main problem associated with symmetric encryption is the secure distribution of the secret key among the communicating parties. If an attacker compromises with the secret key, it could jeopardize the entire encryption system. If the key is known to all, hackers or no authorized people, then they can decrypt the encrypted messages and jeopardize communication security. Despite this challenge, symmetric cryptography remains a pillar of modern cryptography, particularly in situations where speed, efficiency, and security are imperative.

The secrecy of the encrypted data is vulnerable to compromise. Even though symmetric encryption works well for one-to-one communication, it might not be appropriate in situations where there are several participants or when there are intricate key management needs. To overcome these difficulties, it is frequently coupled with other cryptographic methods, such as hybrid encryption or key exchange protocols [3]. The confidentiality of the encoded data is at risk. Despite the effectiveness of symmetric encryption in individual conversations, it may not be suitable in scenarios with multiple participants or complex key administration requirements [6]. It is often combined with alternative cryptographic techniques, like hybrid encryption or key exchange protocols, to tackle these challenges. Each user will generate a key pairing that includes public and private keys as part of the asymmetric encryption process. The public key is given to anyone wishing to communicate securely with the key owner, while the key owner keeps a private key secret. Public-key encryption can be used in many applications, including secure communications (HTTPS for web browsing), digital signatures, email encryption (PGP, S/MIME), and Secure File-Transfer Protocols (SSH, SFTP). Its proficiency in enabling secure communication and ensuring data integrity verification positions it as a cornerstone of modern cryptographic systems [3] consisting of public and private keys.

1.1. Advanced Encryption Standard (AES)

The AES symmetric encryption algorithm uses the same cryptographic key for both encryption and decryption. AES performs multiple substitution, transposition, and mixing rounds for enhanced security while improving the actual data's toughness rather than performing one round of encryption. Compared to typical symmetric encryption, AES operates on data in fixed-size blocks rather than encrypting the entire message simultaneously.

The No. of encryption rounds in a symmetric key algorithm is determined by the key length. AES-128 (128-bit key) executes 10 iterations, AES-192 (192-bit key) executes 12 iterations, and AES-256 (256-bit key) executes 14 iterations.

The basic idea behind each encryption round is to secure the data through complex transformations such that it is more resistant to cryptanalytic attacks. The decryption process is very similar to the encryption process, except the steps are performed reversely to allow the recipient to systematically un-wrap the encryption layers and return the original plaintext after utilizing the same key. This process establishes the confidentiality, integrity, and protection of data in all types of communications and applications.

Different security needs are met with various lengths of AES keys, with the lengths being 128 bits, 192 bits, and 256 bits. Where longer keys offer increased protection but require

more computational resources [5], as you gain a deeper understanding of encryption, you will realize the importance of the Advanced Encryption Standard (AES) in preserving data integrity and privacy in the modern digital era. AES's blend of robust security and effectiveness forms a powerful defense against malicious individuals while giving users smooth data protection throughout numerous applications.

1.1.1. AES-128

AES encryption is being used with a 128-bit key length. The encryption process will consist of 10 rounds. It maintains security and performance, making it perfect for protecting classified information at the secret level.

1.1.2. AES-192

AES encryption is being used with a 192-bit key length. It consists of 6 columns, each containing 32 bits, and requires 12 rounds during encryption. It maintains security and computational speed.

1.1.3. AES-256

AES encryption is being used with a 256-bit key length. This is equivalent to 8 sets of 32-bit columns. The key comprises 8 words, with each word being 32 bits in length. The encryption process consists of 14 rounds. It provides the most powerful encryption, guaranteeing strong data security.

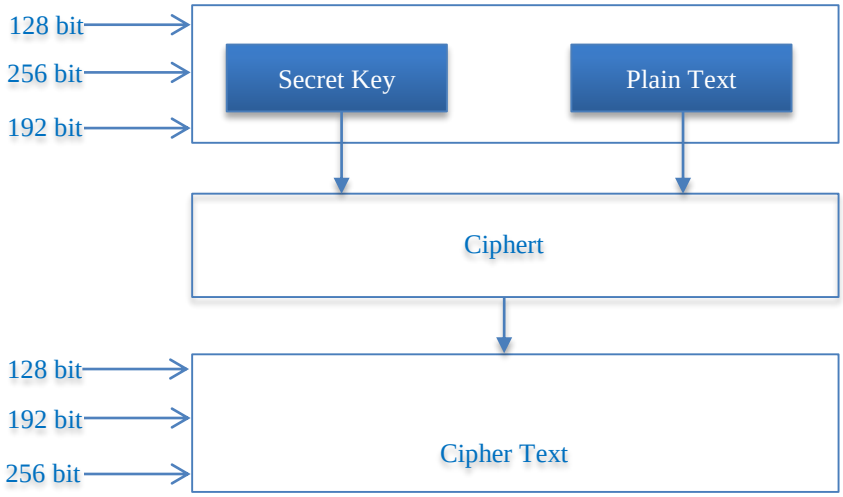

Fig. 1 Advanced encryption standard architecture

Figure 1 shows that AES is a 128-bit encryption algorithm designed for business applications, using 192 and 256-bit keys for encryption [6]. It is highly effective in 128-bit frames and impervious to all attacks except beast compulsion, which uses all possible combinations.

1.2. Rivest Shamir Adleman

The RSA encryption method is known as an algorithm. With algorithms, distinct keys are used to encrypt and decrypt data between the sender and recipient. The private key is for decrypting, and the public key is used to encrypt. Only a public key can decrypt data. Even though the public and private keys are related, it's not feasible to derive the key from one. While the public key is publicly known, the private key remains confidential and accessible to its owner.

This setup allows only the recipient's key to send messages to anyone [7]. Only by using a private key can the recipient decipher the data. In RSA cryptographic encryption, data transmission is done using keys. These keys are associated with numbers with RSA, such as 256,512,1024,2048,4096. These represent the length of the keys in bits, and they represent the security level of RSA encryption.

1.2.1. RSA-256

In this data, encryption will be done by the key to 256 bits in length. It's relatively small and considered to offer weak security, but it is suitable for lightweight applications or scenarios where very high security is not required.

1.2.2. RSA-512

The RSA encryption has a key length of 512 bits. It provides more security when compared to RSA-256, but it is considered weak as of today's scenario.

1.2.3. RSA-1024

This is RSA encryption with a 1024-bit key length. Although it was once widely utilized, it is now thought to provide only modest security. In fact, because RSA-1024 is prone to attacks, it is advised against utilizing it for new applications.

1.2.4. RSA-2048

This refers to RSA encryption with a key of length 2048. Right now, RSA-2048 is the most widely utilized key length. It is generally regarded as secure for many applications and delivers a substantially greater level of security than RSA-1024.

1.2.5. RSA-4096

This refers to 4096-bit key-length RSA encryption. Compared to RSA-2048, it provides even greater security but at an increase of greater computational complexity and longer key generation times. Applications requiring very high security, including those in government or military systems, frequently use RSA-4096.

Longer key lengths might be needed to maintain the same degree of security as cryptographic techniques and processing power grow [7]. The key lengths employed in cryptographic systems must thus remain examined and revised on a regular basis to respond to new security risks.

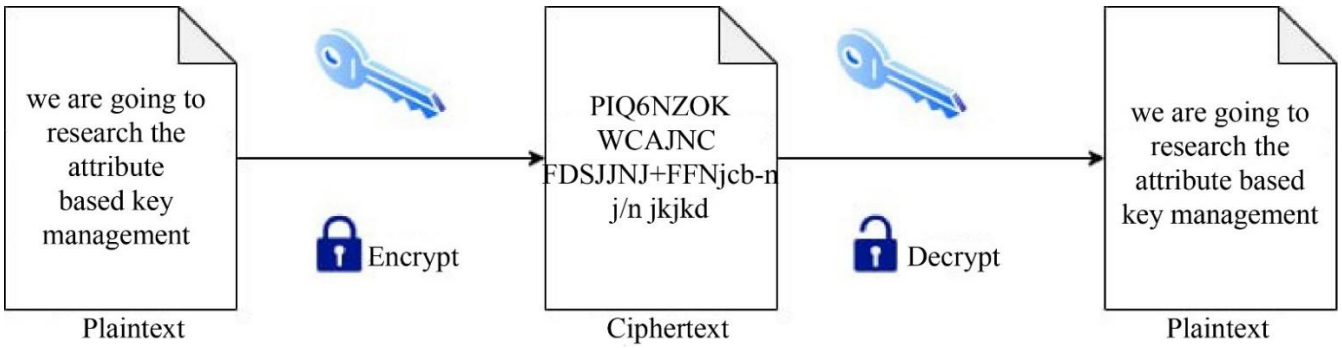

Fig. 2 Rivest shamir adleman architecture

Figure 2 shows the architecture of the RSA algorithm process. It shows how encryption and decryption will be processed in cryptography. It shows how the data is transferred to the recipient after encryption and deciphered by the recipient using the secret key.

1.3. Key Management System (KMS)

The KMS serves as a guardian for keys, ensuring their handling for encryption and decryption processes. It oversees tasks such as rotation, exchange, storage, access control, and security evaluations. By managing and associating keys appropriately, KMS plays a role in protecting information from cyber threats by facilitating the secure sharing of keys with authorized entities and systems [9]. A key management system has different aspects like key Generation, Distribution, Storage, Rotation, Access Control, Logging and Auditing, and Integration.

1.3.1. Key Generation

In KMS, the keys are generated for encryption, decryption, and authenticating users. These create cryptographic keys that are used in security protocols.

1.3.2. Key Distribution

After the keys are generated, the KMS must distribute cryptographic keys securely to the actual users. The keys are used after the authentication, and approved entities have only access to the keys.

1.3.3. Key Storage

In KMS, the cryptographic keys are secured in the storage. It also shows that the keys are used only by authorized people and are secure from unauthorized access. They are to be stored using secure storage mechanisms.

1.3.4. Key Rotation

For security, the keys had to be maintained over time. It is very important to rotate the keys regularly to minimize catastrophic failure between the keys. This updates keys without disrupting operations.

1.3.5. Access Control

The KMS tightly controls key access based on defined roles, permissions, and policies, preventing unauthorized use of sensitive cryptographic material. Only approved individuals and systems can use the keys.

1.3.6. Logging and Auditing

The KMS logs and audits key usage, access attempts, and relevant activities, providing visibility into key management operations to detect potential incidents or violations.

1.3.7. Integration

The KMS can integrate with various applications, databases, and components via APIs, SDKs, or plug-ins to provide seamless encryption and key-management capabilities.

1.3.8. KM Process

The KMS is essentially the central repository of all the activities that pertain to cryptographic keys in relation to an organization's security infrastructure. It is basically a high-tech "vault" that safely produces, stores, disseminates, and administers keys often used in decrypting and encrypting confidential data, digital signatures, and authentication. Specifically, a KMS is built on the complex issue of safely administering cryptographic keys at any point in their lifecycle. This encompasses the generation of the keys themselves, ensuring that they are secure from unauthorized

access, changing them regularly to protect against continuing risk, and distributing them safely to suitable people or systems. Key generation is one of the basic methods that a KMS performs when it typically offers solid algorithms to support building cryptographic keys imbued with a certain level of randomness that is hard to identify using brute force approaches.

KMS enforces access control mechanisms that deny unauthorized cryptographic key access by implementing the concept of least privilege. Properly authenticated users with appropriate permissions can use the system to access the absolute minimum number of keys required for their work. In addition, a high-quality KMS has auditing and compliance features that log all operations associated with key management. As a result, it can adhere to relevant regulatory requirements and support forensic investigations of data security incidents. Interoperability and integration are also critical for proper functioning in an IT environment. As such, a KMS must be able to integrate with all relevant systems, applications, and cryptographic protocols to be easily compatible and deployable.

Lastly, high availability and disaster recovery measures are to be implemented for continuous operations. Incorporating redundancy, safe procedures, and disaster response strategies is crucial to mitigating disruptions and data loss in the event of hardware malfunctions or unforeseen circumstances. Essentially, a critical component of security lies in the management system, as it offers the essential framework and resources for safeguarding confidential data, complying with regulations, and upholding the authenticity of digital interactions within a company.

The information in the paper is arranged as follows: A thorough survey of the literature on various key encryption techniques is provided in Section 2. In the third section, we go over the existing approach to the attribute-based key management encryption algorithm and present our suggested approach, which makes use of well-known cryptographic protocols and techniques like Secure Shell (SSH) and Transport Layer Security (TLS) to protect data in cloud computing environments. We go into great depth on the encryption procedure. The findings and a comparative study of our suggested methodology are shown in Section 4. Section 5 offers a comprehensive elucidation of the study paper's outcomes and concluding remarks.

2. Literature Review

The author’s research paper titled Blockchain-Based Privacy Protection Third-Party Service Transparency Framework introduces the TAB framework, which uses blockchain technology to verify third-party clarity and transparency. Schools and institutions. Reliability Privacy Protection Enforcement Facility. TAB uses the Ethereum blockchain and innovative new contracts to support corporate investigations and punish bad behavior. Clinical evaluation of the Rinkeby test network demonstrates TAB's performance and safety assurance.

The author of the research paper titled Eddystone-EID: Security and Identity Enhancement for BLE Beacons [2] proposed the Eddystone-EID protocol to solve the identified privacy and security issues of BLE Beacons in IoT applications. Eddystone-EID creates a secure connection between beacons and their owners, emphasizing privacy, security, and low power consumption, reducing privacy concerns associated with broadcast beacon identity and telemetry data.

The author of the research book TNGuard: Protecting Security SDN-Based Tenant Network for IoT [3] focuses on protecting Tenant Networks (TN) in SDN-based cloud environment security. TNGuard improves tenants' security and solves flaws in data security and access control by using TN abstraction and limiting cloud management permissions.

The article by the author of the research book Research on Blockchain-Based Business Research [4] explores the use of blockchain technology in the management of business transactions. It compares it with traditional cloud storage networks. It discusses Blockchain's role in ensuring data security and integrity, highlights solutions such as SIA, File coin, and Store, and addresses security issues and solutions in shared storage. Access Control and Data Sharing for SCADA in IoT Systems is suggested by the author of a research paper called Secure Retrievable Fine-Graining [5], which solves the security risk of SCADA systems in the IIoT environment. The solution manages operational vulnerabilities while improving data confidentiality and access control using digital signature technology.

An article written by the author of a research book titled The Internet of Things (IoT) Forensic Investigations: Challenges, Methods, and Open Issues [6] addresses IoT-based research issues, including legal, privacy, and security. Cloud security questions. It explores privacy-preserving data storage systems, blockchain-based evidence-based justice solutions, and new models such as Forensics as a Service (FaaS) and data reduction technology.

The research paper titled Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads by Authors [7] evaluates the performance of QUIC across various application workloads compared to traditional TLS/TCP. It highlights QUIC's advantages in terms of connection times, latency, throughput, and content delivery efficiency.

The research paper titled IoT Sensor Initiated Healthcare Data Security by Authors [8] addresses security risks in healthcare data transmission via IoT. It proposes an encryption solution embedded in IoT sensor devices for end-to-end security.

The article's title is a lightweight authentication encryption scheme based on advertiser-subscriber communication for the Internet of Things Symmetric Key Encryption Security.

The paper "Blockchain-Based Secure Storage and Trusted Data Sharing Solution for the Internet of Things Environment" [10] proposes the IoTCain model, which merges the Ethereum blockchain and the Interplanetary File System (IPFS) to enable secure and decentralized data storage in the IoT ecosystem. This merging provides enhanced security, data integrity, and trust in distributed, interconnected environments. The research [11] on "Secure Outsourcing and Sharing of Cloud Data Using a User-Side Encrypted File System" demonstrates OUTfs, a user-side encryption framework that employs identity-based encryption to enhance the security, privacy, and efficiency of cloud data sharing. The proposed solution seeks to protect sensitive data while preserving seamless access control and confidentiality in cloud environments. Despite being an efficient system for maintaining data integrity and providing strong encryption, some drawbacks associated with OUTfs are low efficiency during higher load levels, scalability problems, and limitations in supporting mixed workloads.

The authors in the research paper “Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption” [12] present a framework for secure data sharing within a cloud computing environment using revocable-storage identity-based encryption. The significance of this approach lies in the fine-grained access control and potential for revoking access to a user, should that be required. The challenge is to update the ciphertext while keeping encryption secure in the presence of security risks, which could potentially compromise the confidentiality and/or integrity of data.

The authors explain the study “Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data” [13], which has an emphasis on the strength and efficiency of the AES encryption algorithm in terms of secure data protection. AES is thought to have put itself in the ranks of the fastest, most reliable, and most resistant to most forms of cryptographic attacks. However, using shorter key lengths can enable the potential for different attacks on the algorithm, usually through brute-force attacks, which, in turn, indicate a need to use longer key sizes such as AES-256.

The authors of the study “Design of a Novel Hardware Architecture for Fully Homomorphic Encryption (FHE) Algorithms in FPGA for Real-Time Data in Cloud Computing” [14] discuss the author’s development of a novel or specialized hardware design of FHE effectuated on a Field Programmable Gate Array (FPGA) platform. This architecture seeks to improve computation efficiency and enable low-latency, high-throughput encryption operations. This study advances the practicality of real-time processing of data in the cloud via FPGAs while enabling security and performance enhancement. One challenge might be the large storage requirement for key management, but this can be easily solved with modern computers available today.

The authors' research paper called Innovative Approach for Securing Cloud Data; AES Encryption and Blockchain Key Management [15] suggests a strategy that integrates AES encryption and blockchain key management to boost cloud data security. They point out resource usage and performance issues as concerns.

The research paper Enhancing Encrypted Data Sharing in the Public Cloud through Identity Based Encryption Transformation by Authors [16] presents IBET as a solution for flexible data sharing in the cloud. Nonetheless, centrally managing keys could pose scalability issues. Increase burden.

The research paper named Enhancing End-to-End Security in Cloud/Edge Computing through Public Key Encryption Resilient to Randomness Attacks by the authors [17] delves into the topic of public key encryption resilience against randomness attacks, focusing on bolstering security measures in cloud and edge computing. Nevertheless, it highlights that the security of encryption methods could be influenced by assumptions.

The paper by Zhiwei Wang, Longwen Lan, and Siuming You [18] introduces a modified key management system (UOKM) based on chameleon hashing, which solves key management problems in cloud storage. Their strategy uses chameleon hashing to achieve faster updates without compromising security, thus eliminating the overhead of public key authentication and security channels. This method improves the security and efficiency of key management in large storage systems and benefits the cloud security industry.

The paper by Mohammed Y. Shakor [15] proposes an effective cloud information security methodology through dynamic AES encryption and blockchain key management. This cutting-edge solution addresses the shortcomings of conventional encryption and key storage by generating new AES keys for every single file, therefore increasing data-level security, as well as providing blockchain-based tools to securely store and manage metadata. These features overcome many limitations and enhance the cloud security framework by promising strong encryption and effortless key management, in addition to preventing intrusions to safeguard users' data in cloud storage systems. Meng's research also presents the certificate-free 0-RTT anonymity, AKA protocol for establishing a fast, secure, and efficient cloud environment. As a Zero-Round Trip Time (0-RTT) protocol, it optimizes secure channel establishment to reduce latency while maximizing real-time security. By eliminating the

reliance on certificates to validate public keys, it also resolves authentication management concerns. Finally, the system solidifies security by protecting user privacy and preventing unconsented randomness, improving cloud-based service's overall resilience and efficiency. The study helps to improve the security and efficiency of cloud communications and solves the fundamental problems in the security of public communications between cloud users and service providers.

The paper by Bojjagani, Reddy, Anuradha, Rao, Reddy, and Khan [20] addresses the security issues of the Internet of Vehicles (IoV) for smart transportation. AKAP-IoV systems provide secure communication, coordination and critical control. The research includes tests, security analysis, and comparisons of AKAP-IoV performance and security features to ensure clear protection against cyber threats.

Table 1. Literature review on existing methodologies

S.No.	Authors	Title	Applied Methodology	Drawbacks
1	Runhua Xu, Chao Li, James B.D. Joshi [1]	Blockchain Based Privacy Protection Third-Party Service Transparency Framework.	Ethereum blockchain, innovative contracts.	The TAB framework may face scalability challenges as the blockchain network grows.
2	David L, Avinatan H, Yossi M, Yung, and Ziv [2]	Eddystone-EID: Security and Identity Enhancement for BLE Beacons.	Eddystone-EID protocol.	The Eddystone-EID protocol's implementation complexity may pose integration challenges for existing BLE beacon systems.
3	Zirong Huang	TNGuard: Protecting Security SDN-Based Tenant Network for IoT.	TN abstraction, limited cloud management permissions.	TNGuard's effectiveness may depend on the scalability of SDN-based networks.
4	Ibtisam Ehsan, Irfan Khalid Ayman Hallel Al-Ani Muhammad, [3]	Research on Blockchain-Based Business Research.	Blockchain technology comparison with traditional cloud storage.	The comparison may overlook specific use-case differences that affect performance and security.
5	Weiting Zhang, Hanyi Zhang, Liming Fang, Chunpeng Liu Zhe and Ge [4]	Access Control and Data Sharing for SCADA in IIoT Systems.	Digital signature technology.	The reliance on digital signatures may introduce computational overheads in SCADA systems.
6	Stoyanova Maria, Nikoloudakis Yannis, Panagiotakis Spyridon, Pallis and K. Markakis [5]	Internet of Things (IoT) Forensic Investigations: Challenges, Methods, and Open Issues.	Privacy-preserving data storage systems, blockchain-based justice solutions.	The scalability and real-time performance of forensic investigations using these methods may be limited.
7	Tanya, V Bajpai [6]	Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads.	Comparative analysis of QUIC with traditional TLS/TCP.	The evaluation may not cover all possible scenarios and may not consider specific network conditions.
8	Kedir Besher, Subah Zareen, Zamshed Ali Mohammed [7]	IoT Sensor Initiated Healthcare Data Security.	Encryption solution embedded in IoT sensor devices.	The encryption solution's overhead on IoT sensor devices may impact their performance or battery life.
9	Abebe, Reda, Naveen Ch, Mahmood, Noor Zaman, Nam Yunyoung [8]	A Lightweight Authentication Encryption Scheme Based on	Symmetric key encryption.	The symmetric key encryption scheme may face key distribution challenges and scalability issues in large IoT networks.

		Advertiser-Subscriber Communication for the Internet of Things.		
10	Zia Ullah and Kolay [9]	Blockchain-based Secure Storage and Trusted Data Sharing Solution for the Internet of Things Environment.	IoTCain model integration of Ethereum blockchain and IPFS.	The integration of multiple technologies may introduce interoperability challenges and increase complexity.
11	Zhiwei Wang, Longwen Lan and Siuming You	Modified Key Management System (UOKM) based on Chameleon Hashing for Cloud Storage.	Chameleon hashing faster updates without compromising security; Eliminates overhead of public key authentication and security channel.	Lack of scalability for extremely large storage systems; Potential vulnerabilities in the chameleon hashing algorithm.
12	Shakor, Mohammed Y. [14]	Effective Data Security in the Cloud using Dynamic AES Encryption and Blockchain Key Management.	Dynamic AES encryption, Blockchain for key management, and Improved data-level security.	Overhead in key generation and management due to blockchain integration; Complexity in blockchain maintenance and scalability.
13	Meng, Xinyu, Kang and Burong [15]	Certificate-free 0-RTT Anonymity AKA Protocol for Secure Cloud Environment.	Zero Round-Trip Time (0-RTT) protocol for faster channel establishment; Eliminates the need for certificates; Provides user privacy and randomness prevention.	Vulnerability to replay attacks due to lack of session identifiers; Potential latency issues in establishing secure channels.
14	Bojjagani,Anuradha, and Khan [20]	Security Issues in Internet of Vehicles (IoV) for Smart Transportation.	AKAP-IoV systems for secure communication and control; Performance and security analysis.	Limited compatibility with legacy IoV systems; Potential resource overhead in implementing AKAP-IoV.

In the above Table, 1 is the summary of the literature on current techniques for improving security in cloud computing environments. It compares several strategies, such as the Ethereum blockchain, digital signature technology, privacy-preserving data storage systems, dynamic AES encryption, AKAP-IoV systems for secure communication and control, Zero Round-Trip Time (0-RTT) protocol, and more.

3. Proposed Methodology

In this project, Symmetric keys and Asymmetric keys are used with the Key Management Process and by using an Attribute-based key management encryption algorithm. As we know, Key management is paramount in rotation, generation, and revocation of symmetric and asymmetric keys because it will ensure the confidentiality of the data encrypted by safeguarding the encryption key. It also prevents data integrity by denying unauthorized data modification in encrypted data. Key management also decreases the hazard of information violations through our encrypted data, which is safe from other unauthorized access.

Overall, our proposed methodology states that key management plays an important role in the security and reliability of encrypted data in both asymmetric and symmetric encryption algorithms. To further process the data, the encryption requirements and sensitivity levels of the data need to be known.

The above-proposed methodology determines the appropriate key length because a symmetric key supports lengths of 128,196, and 256. Key lengths must remain chosen based on the security level and the requirements. A secure

method should be implemented for the random key generation for symmetric keys; it can be possible by utilizing the Cryptographically Secure Pseudo-random Number Generators (CSPRNGs) to ensure the randomness and unpredictability of generated keys. The proposed methodology will generate symmetric keys by designing a key derivation process from random data. There is a need to establish a secure storage system to store the symmetric keys securely from unauthorized access. Asymmetric key pairs consist of two key pairs: public key and private key. Asymmetric keys are typically generated as prime number pairs (n, e) for the public key and (n, d) for the private key. Determining key lengths for asymmetric keys based on security requirements and choosing longer key lengths for higher security but will impact the performance of the keys. Common asymmetric key lengths range from 1024 to 4096 bits. Implement the same storage method for asymmetric keys as we did for symmetric keys to prevent the encrypted data from unauthorized access. We need to utilize the established cryptographic protocols and techniques such as Transport Layer Security (TLS) or Secure Shell (SSH) to implement secure communication channels between both keys for key exchange. Follow the key rotation policies for symmetric and asymmetric keys for secure key rotation. Regularly replace the existing keys with the new keys that were created for key rotation. Symmetric keys should regularly rotate every 90-180 days; asymmetric keys should not be rotated very frequently because it leads to computational overhead, but there is a need to rotate them annually. Key rotation mitigates the impact of the key compromise and ensures long-term security.

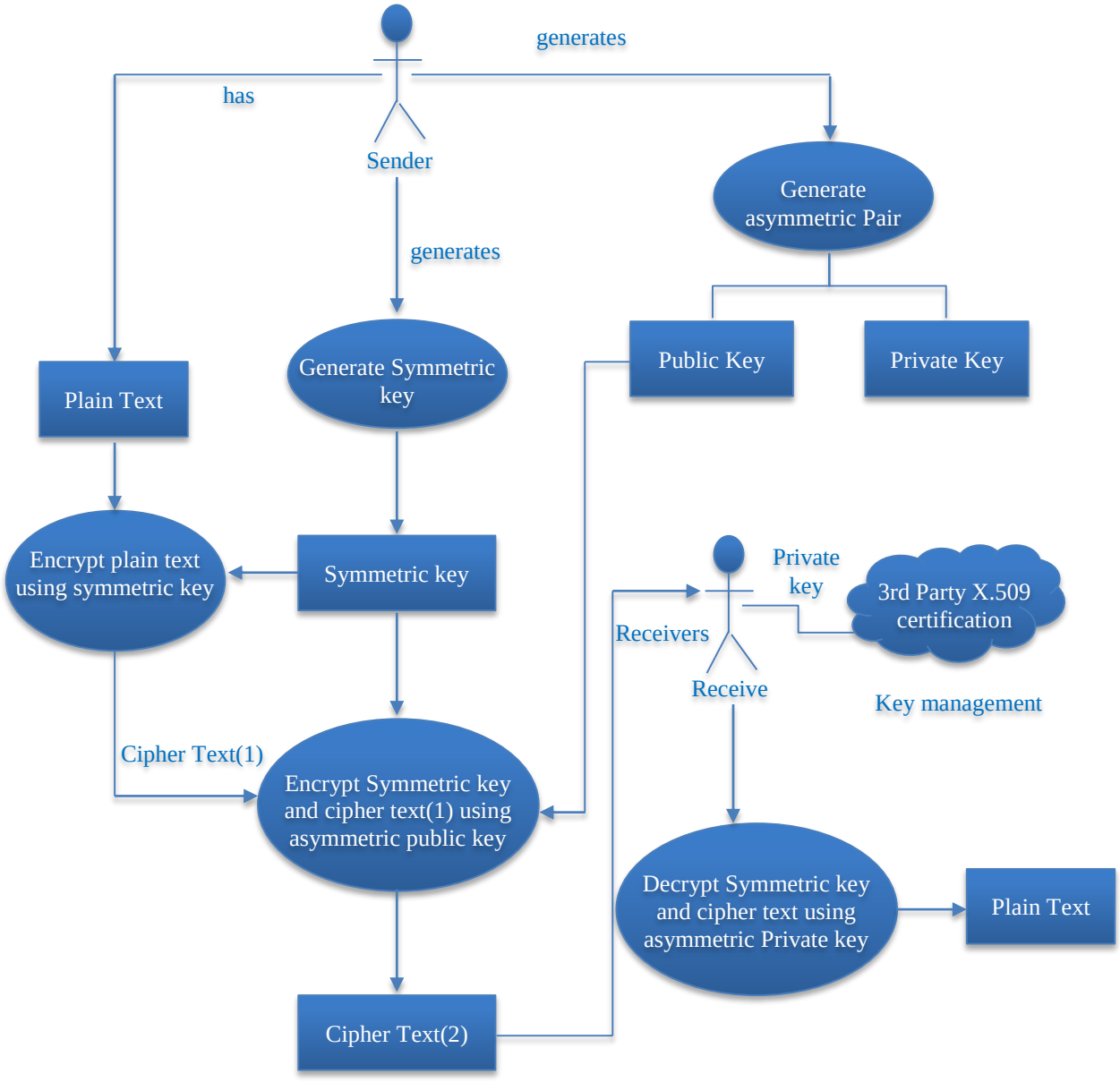

Fig. 3 Proposed algorithm of Attribute-Based key management encryption algorithm

In Figure 3, mechanisms for key expiration and revocation need to be developed to maintain the security of cryptographic systems. To revoke the keys which are no longer needed by doing that can protect them from unauthorized access. Implementing secure key destruction procedures will ensure irrecoverable deletion of keys once they are no longer needed. Utilizing cryptographic erasure techniques includes further safety by rendering keys unreadable and irrecoverable, thereby protecting sensible data from unauthorized access or misuse. Ensuring robust auditing and monitoring capabilities is essential for tracking key usage, access attempts, and any changes to key configurations. By setting up alerts, it can promptly detect unauthorized key access or suspicious activities, thereby enhancing our security measures. Additionally, maintaining comprehensive audit logs ensures compliance and facilitates forensic analysis in the event of security incidents. These measures collectively contribute to safeguarding the integrity and confidentiality of cryptographic keys, bolstering overall system security.

3.1. Proposed Algorithm

- Encryption of plain text $C=E_{SYMMETRIC}(P, K_{SYMMETRIC})$
- Symmetric key generation using Key management process $K_{SYMMETRIC}$ = KeygenerationProcess()
- Asymmetric key pair generation with various key sizes:
  1. $(K_{ASYMMETRIC(public)}, K_{ASYMMETRIC(private)})_{256}$ = AsymmetricKeyGenerationProcess(256)
  2. $(K_{ASYMMETRIC(public)}, K_{ASYMMETRIC(private)})_{512}$ = AsymmetricKeyGenerationProcess((512)
  3. $(K_{ASYMMETRIC(public)}, K_{ASYMMETRIC(private)})_{1024}$ = AsymmetricKeyGenerationProcess((1024)
  4. $(K_{ASYMMETRIC(public)}, K_{ASYMMETRIC(private)})_{2048}$ = AsymmetricKeyGenerationProcess((2048)
  5. $(K_{ASYMMETRIC(public)}, K_{ASYMMETRIC(private)})_{4096}$ = AsymmetricKeyGenerationProcess((4096)
- Encryption of symmetric key with recipient's asymmetric public key for each key size:
  1. $C_{SYMMETRIC(256)} = E_{ASYMMETRIC}(K_{SYMMETRIC}, K_{ASYMMETRIC(public)})_{256}$
  2. $C_{SYMMETRIC(512)} = E_{ASYMMETRIC}(K_{SYMMETRIC}, K_{ASYMMETRIC(public)})_{512}$
  3. $C_{SYMMETRIC(1024)} = E_{ASYMMETRIC}(K_{SYMMETRIC}, K_{ASYMMETRIC(public)})_{1024}$
  4. $C_{SYMMETRIC(2048)} = E_{ASYMMETRIC}(K_{SYMMETRIC}, K_{ASYMMETRIC(public)})_{2048}$
  5. $C_{SYMMETRIC(4096)} = E_{ASYMMETRIC}(K_{SYMMETRIC}, K_{ASYMMETRIC(public)})_{4096}$
- Sending encrypted symmetric key and asymmetric-encrypted message to the recipient.
- Decryption of symmetric key using recipient's asymmetric private key: $K_{SYMMETRIC} = D_{ASYMMETRIC}(C_{SYMMETRIC}, K_{ASYMMETRIC(private)})$
- Decryption of symmetric-encrypted message: $P = D_{SYMMETRIC}(C, K_{SYMMETRIC})$
- Rotation of symmetric keys: $K_{SYMMETRIC}$ = RotateKeys($K_{SYMMETRIC}$)
- Key revocation mechanism for assymtric keys if compromised.
- Monitor key usage, access attempts, and changes to key configurations:
  - Implementing monitoring and auditing capabilities.
- Establish an alert system for unauthorized access.
- Generate audit logs for compliance and forensic analysis.

Certainly, let's define the names for the variables used in the algorithm:

1. P - Plain text message.
2. C - Encrypted message.
3. $K_{SYMMETRIC}$ - Symmetric key used for encrypting and decrypting the message.
4. $E_{SYMMETRIC}$ – Symmetric encryption algorithm
5. $D_{SYMMETRIC}$ – Symmetric decryption algorithm
6. $K_{ASYMMETRIC(public)}$ – Asymmetric Public key
7. $K_{ASYMMETRIC(private)}$ – Asymmetric Private key
8. $E_{ASYMMETRIC}$ – Asymmetric Encryption algorithm
9. $D_{ASYMMETRIC}$ – Asymmetric Decryption algorithm
10. $C_{SYMMETRIC(256)}$ - Encrypted Symmetric key using RSA with a 256-bit key.

11. $C_{SYMMETRIC\ (512)}$ - Encrypted Symmetric key using RSA with a 512-bit key.
12. $C_{SYMMETRIC\ (1024)}$ - Encrypted Symmetric key using RSA with a 1024-bit key.
13. $C_{SYMMETRIC\ (2048)}$ - Encrypted Symmetric key using RSA with a 2048-bit key.
14. $C_{SYMMETRIC\ (4098)}$ - Encrypted Symmetric key using RSA with a 4098-bit key.
15. KeyGenerationProcess() - Process to generate AES keys.
16. AsymmetricKeyGenerationProcess(*n*) - Process to generate asymmetric key pairs with key size *n* bits.
17. RotateKeys($K_{SYMMETRIC}$) - Process to rotate symmetric keys for mitigating the risk of key compromise.
18. Monitoring, auditing, and alert mechanisms are abstract concepts without specific variable names.

4. Results and Discussion

The algorithm utilizes a hybrid encryption method in which both Symmetric Key (AES) and Asymmetric Key (RSA) cryptography are utilized. Symmetric keys are generated for encryption, whereas asymmetric key pair sizes produce strong encryption. Encrypted symmetric keys are designed for the recipient's key sizes, guaranteeing secure key exchange. While decryption uses private keys, recipients will rapidly access their symmetric keys, which allows for effortless decryption.

Table 2 compares symmetric (AES) and asymmetric (RSA) algorithms with our proposed algorithm using an attribute-based key encryption algorithm.

Overall, using our proposed methodology can provide us with a better throughput of computation time when compared to AES and RSA algorithms.

Table 2. Computation time of proposed algorithm

Data (KB)	AES (Time (Sec))	RSA (Time (Sec))	Proposed (Time (Sec))
128	2.2	-	1.79
192	2.41	-	1.81
256	2.51	-	2.02
512	-	9.41	2.11
1024	-	10.52	2.42
2048	-	11.39	2.53
3072	-	16.23	2.62
4096	-	24.41	2.74

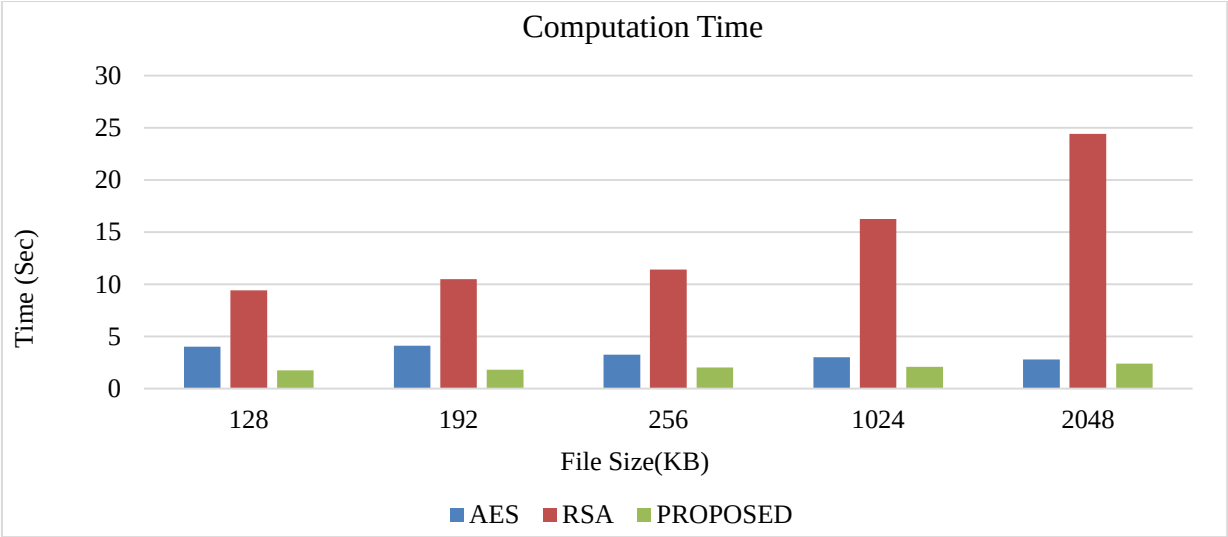

Fig. 4 Computation time comparison

Figure 4 is a graph of comparison between AES, RSA, and our Proposed algorithms. There is a difference in computation time among AES, RSA, and our proposed algorithms.

The computation time of RSA is higher when compared to AES and the proposed algorithm, and the computation time of AES is less than that of RSA, but AES has more computation time than our proposed algorithm. So, from analysis, using our Proposed algorithm, the throughput of computation time can be achieved in less time and in a more efficient way.

5. Conclusion

This study aims to investigate the increasing cloud security problem by assessing the current literature. A practical solution was provided by implementing an attribute-based key management system based on key management services. Performance testing was conducted by measuring the File Size (KB) and Time (sec) to show that this approach was superior to standard encryption algorithms (AES, RSA, DES) and homomorphic encryption. The study provides strong end-to-end encryption, indicating that the mechanism enhances the security of sensitive data stored in the cloud.

References

[1] Runhua Xu, Chao Li, and James Joshi, "Blockchain-Based Transparency Framework for Privacy Preserving Third-Party Services," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 3, pp. 2302-2313, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Liron David et al., "Eddystone-EID: Secure and Private Infrastructural Protocol for BLE Beacons," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3877-3889, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Weiqi Dai et al., "TNGuard: Securing IoT Oriented Tenant Networks Based on SDN," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1411-1423, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Muhammad Irfan Khalid et al., "A Comprehensive Survey on Blockchain-Based Decentralized Storage Networks," *IEEE Access*, vol. 11, pp. 10995-11015, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Weiting Zhang et al., "A Secure Revocable Fine-Grained Access Control and Data Sharing Scheme for SCADA in IIoT Systems," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1976-1984, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Maria Stoyanova et al., "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1191-1221, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Tanya Shreedhar et al., "Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1366-1381, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Kedir Mamo Besher, Zareen Subah, and Mohammed Zamshed Ali, "IoT Sensor Initiated Healthcare Data Security," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11977-11982, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Abebe Diro et al., "Lightweight Authenticated-Encryption Scheme for Internet of Things Based on Publish-Subscribe Communication," *IEEE Access*, vol. 8, pp. 60539-60551, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Zia Ullah et al., "Towards Blockchain-Based Secure Storage and Trusted Data Sharing Scheme for IoT Environment," *IEEE Access*, vol. 10, pp. 36978-36994, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Osama Ahmed Khashan, "Secure Outsourcing and Sharing of Cloud Data Using a User-Side Encrypted File System," *IEEE Access*, vol. 8, pp. 210855-210867, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[12] Kwangsu Lee, "Comments on "Secure Data Sharing in Cloud Computing Using Revocable-Storage Identity-Based Encryption," *IEEE Transactions on Cloud Computing*, vol. 8, no. 4, pp. 1299-1300, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ako Muhamad Abdullah, "Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data," *Cryptography and Network Security*, vol. 16, no. 1, pp. 1-12, 2017. [Google Scholar]

[14] Sagarika Behera, and Jhansi Rani Prathuri, "Design of Novel Hardware Architecture for Fully Homomorphic Encryption Algorithms in FPGA for Real-Time Data in Cloud Computing," *IEEE Access*, vol. 10, pp. 131406-131418, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Mohammed Y. Shakor et al., "Dynamic AES Encryption and Blockchain Key Management: A Novel Solution for Cloud Data Security," *IEEE Access*, vol. 12, pp. 26334-26343, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[16] Hua Deng et al., "Identity-Based Encryption Transformation for Flexible Sharing of Encrypted Data in Public Cloud," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3168-3180, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Pengtao Liu, "Public-Key Encryption Secure Against Related Randomness Attacks for Improved End-to-End Security of Cloud/Edge Computing," *IEEE Access*, vol. 8, pp. 16750-16759, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Zhiwei Wang, Longwen Lan, and Siuming Yiu, "Chameleon Hash Based Efficiently Updatable Oblivious Key Management," *IEEE Transactions on Services Computing*, vol. 16, no. 6, pp. 4503-4513, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Xinyu Meng, Lei Zhang, and Burong Kang, "Fast Secure and Anonymous Key Agreement against Bad Randomness for Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1819-1830, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Sriramulu Bojjagani et al., "Secure Authentication and Key Management Protocol for Deployment of Internet of Vehicles (IoV) Concerning Intelligent Transport Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24698-24713, 2022. [CrossRef] [Google Scholar] [Publisher Link]