JITK (JURNAL ILMU PENGETAHUAN DAN TEKNOLOGI KOMPUTER)

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v11i2.7132

CRYPTOGRAPHIC FRAMEWORK FOR CLOUD-BASED DOCUMENT STORAGE USING AES-256 AND SHA-256 HYBRID SYSTEMS

Junaidi Surya[1*], Ahmad Louis[2], Faiza Rini[1], Sri Mulyati[1], Elzas[1]

Faculty of Computer Science, Depertement of Information System[1]
Faculty of Science and Technology, Depertement of Information Technology[2]
University Nurdin Hamzah, Jambi, Indonesia[1,2]
https://unh.ac.id/[1,2]
junaidis10@email.com*, louis124fi@gmail.com, faizarini201104@gmail.com,
mulyati.sri52@gmail.com, ethas78@gmail.com

(*) Corresponding Author
(Responsible for the Quality of Paper Content)

Abstract— *Cloud-based document storage offers significant flexibility but faces security challenges such as the risk of data leaks and illegal modifications. The study proposes a cryptographic framework using a combination of Advanced Encryption Standard (AES)-256 for confidential encryption and Secure Hash Algorithm (SHA)-256 for cloud storage-based document integrity verification. The system was developed with an experimental approach, implemented in application prototypes, and tested on a wide range of file sizes from as small as < 1 mb, 10 mb to 100 mb showing greater efficiency than Rivest-Shamir-Adleman (RSA) and elliptical curve cryptography (ECC). To improve security, a distributed key management scheme and password-based user authentication were added. The encryption system will be tested on Google Drive, One Drive, and mega cloud platforms and evaluated through a series of performance and security tests combined with on-premises personal computer (PC) systems. This framework provides a practical solution for secure document storage in the cloud with a balance between security, performance, and ease of use. This research reinforces the urgency of applying modern cryptography in dealing with the risk of data leakage in public cloud services, and can be adopted as a security and efficiency model and solution for individuals, as well as government and private offices that use cloud storage as a storage base for important documents such as Decrees, Securities, certificates, diplomas and other important data.*

Keywords*: AES-256, Cloud Drive, Cryptography, Data Integrity, SHA-256*

Intisari— *Penelitian ini mengusulkan kerangka kerja kriptografi berbasis AES-256 dan SHA-256 secara hibrid untuk meningkatkan keamanan dokumen penting yang disimpan berbasis sistem cloud storage. AES-256 digunakan untuk mengenkripsi dokumen secara end-to-end, sedangkan SHA-256 digunakan menjamin integritas data berupa file melalui verifikasi sidik jari digital. Hasil pengujian menunjukkan bahwa AES-256 memberikan waktu enkripsi/dekripsi yang lebih efisien dibandingkan RSA dan ECC pada file ukuran besar, sementara SHA-256 terbukti menjaga integritas data secara konsisten dan berhasil mendeteksi perubahan data sekecil apa pun dengan akurasi 100%. Pengujian juga menunjukkan sistem tahan terhadap serangan brute-force dan man-in-the-middle. Dengan desain modular AES-256 dan SHA-256, dapat dijadikan solusi dan dapat diterapkan oleh individu, sektor swasta, maupun pemerintah yang menggunakan penyimpan cloud sebagai model perlindungan data cadangan secara efesien.*

Kata Kunci*: AES-256, Penyimpanan Awan, Kriptografi, Integritas Data, SHA-256*

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480 /jitk.v11i2.7132

JITK (JURNAL ILMU PENGETAHUAN DAN TEKNOLOGI KOMPUTER)

INTRODUCTION

Digital transformation has fundamentally changed the way organizations and individuals manage information, with *cloud storage* emerging as a key pillar in document storage and exchange globally [1][2], [3]. The advantages offered, such as ease of data sharing, *real-time collaboration*, and dynamic resource elasticity, have driven the mass adoption of these platforms across various sectors[4]. However, the ease of access and centralization of this data inherently creates a security paradox. On the one hand, data becomes more accessible for productivity purposes; On the other hand, it becomes a more concentrated target and vulnerable to cyber threats [5]. Documents stored in the *cloud* often contain sensitive information ranging from personal data, company trade secrets, to strategic government records *whose confidentiality* and integrity must be maintained absolutely [4], [6].

A number of widely publicized cybersecurity incidents highlight this vulnerability. Data leaks, both caused by external attacks and *insider threats*, as well as illegal access have become common occurrences that harm organizations financially and reputationally. More so, the inability to guarantee that a document is not legally altered can undermine public trust and even lead to serious legal consequences. The main problem faced is that many cloud storage systems rely on server-side protection mechanisms, while data during transmission or even while inside the service provider's infrastructure can be exposed if it is not protected by strong encryption on the client side. The types of data theft, hacking, and file integrity sabotage threats are constantly evolving and evolving, demanding more robust and sophisticated security solutions than just conventional user authentication.

In response to this urgent need, the study aims to develop, implement, and evaluate a comprehensive cryptographic framework to ensure the security of cloud-based document storage [1], [7]. The proposed solution is based on a strategic combination of two globally tested and recognized cryptographic algorithms: the Advanced Encryption Standard with 256-bit keys (AES-256) to ensure the confidentiality of data files through symmetric encryption meaning that they have the same key, and the 256-bit Secure Hash Algorithm (SHA-256) to verify the integrity of encrypted data [3], [8], [9]. The framework is designed with a focus on ease of integration into today's modern cloud storage platforms, so it can be adopted with minimal barriers.

The main contributions of this study can be summarized in three main pillars: (1) the design of a cloud document security system architecture that combines *end-to-end encryption* and hash-based integrity verification; (2) empirical testing of system performance in terms of encryption-decryption process speed, storage usage efficiency, and reliability in detecting the slightest data modification; and (3) evaluation of the system's resilience to relevant real-world attack simulations, including *brute-force* and *man-in-the-middle attacks*. Through this approach, this research not only presents theoretical concepts, but also models that have been proven to be practical and can be adopted as a standard reference by government agencies, industrial sectors, and individual entities that prioritize the security of their digital assets[4].

MATERIALS AND METHODS

This literature review aims to establish a theoretical and contextual foundation for the proposed framework. The discussion began with the cloud storage security landscape, continued with an in-depth analysis of the selected cryptographic technologies (AES-256 and SHA-256), and ended with a review of previous studies to identify existing research gaps.

1. Cloud Storage Security

Today's cloud storage services are highly attractive with their cross-platform data access reliability, high scalability, and operational cost efficiency [10]. With a data storage system model that relies on third-party infrastructure inherently creates significant and complex security gaps. The main security threats in *cloud storage* that the researchers highlighted are:

1) Unauthorized access.
2) Data Leaks
3) Loss of control and visibility.

As a result, relying entirely on the security mechanisms provided by cloud service providers is an inadequate approach. Implementing security with additional standard safeguards, such as **end-to-end *data encryption*** whose keys are managed by the owner of the software data (*Customer-Managed Keys*) and the application **of the principle *of least privilege***, is critical to ensuring the confidentiality, integrity, and availability of data [11].

2. Algorithms and comparison of cryptographic techniques between AES-256, RSA and ECC

Accredited Rank 2 (Sinta 2) based on the Decree of the Dirjen Penguatan RisBang Kemenristekdikti No.225/E/KPT/2022, December 07, 2022. Published by LPPM Universitas Nusa Mandiri

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v11i2.7132

A. **Advanced Encryption Standard (AES)-256.**

The Advanced Encryption Standard (AES) has been recognized as a global industry standard by various agencies, including the U.S. government. Encryption uses AES-256, which uses 256-bit keys, the 4 main pillars in *the Advanced Encryption Standard (AES)* encryption algorithm are *SubBytes, ShiftRows, MixColumns*, and *AddRoundKey*. *SubBytes* override bytes of data, *ShiftRows* shift rows, *MixColumns* mix columns, and *AddRoundKey* combines data with keys[8][12]. The AES-256 algorithm works by breaking plaintext into fixed-sized (128-bit) blocks. Each block then undergoes a series of mathematical transformations (called **rounds**) to convert them into ciphertext. The padding algorithms in AES-256 and PKCS7 can be used in combination. The initial plaintext data will be *padded,* so that it is a multiple of 128 bits. Then, the *padded* data is encrypted using the AES-256 algorithm. This key must be securely shared with the communicating party.

Advantages:

1) Very fast and efficient: Suitable for encrypting large amounts of data, such as files, databases, or streaming connections.
2) Secure: With a 256-bit key size, AES is very difficult to crack. To date, no practical attack has succeeded in breaking AES.
3) Lightweight: Requires relatively low computing resources, making it ideal for devices with limited specifications.

Disadvantages:

1) Key distribution issues: Keys must be shared securely (multi-factor authentication (MFA): a combination of passwords + OTP codes sent via email/phone) before communication begins, which can be challenging.
2) Not ideal for digital signatures: Because it uses a single key, there's no way to prove the origin of the message

B. **Rivest-Shamir-Adleman Encryption** (RSA)

RSA is an asymmetric algorithm that uses a pair of keys: a public key for encryption and a private key for decryption. Public keys can be shared freely, while private keys must be kept confidential.

Advantages:

1) Secure for key exchange: The sender can encrypt data using the recipient's public key, and only the recipient who has the private key can unlock it.
2) Can be used for digital signatures: With a private key, we can use it to sign messages, and others can verify it using the public key. This proves the identity of the sender.

Disadvantages:

1) Very slow**:** So it's inefficient to encrypt large amounts of data.
2) Requires large technology and resources: Large private key sizes (2048-bit or 4096-bit) require high computing power.
3) Vulnerable to quantum computers: The security of RSA depends on the difficulty of factoring large primes. Theoretical quantum computers can solve this problem quickly.

C. Elliptical curve cryptography (ECC)

ECC is an **asymmetric** algorithm that also uses public and private key pairs, but is based on elliptical curve mathematics.

Advantages:

1) Smaller key size: ECC offers the same level of security as RSA, but with a much smaller key size. For example, a 256-bit ECC key is equivalent to a 3072-bit RSA key security.
2) Faster and more efficient: Because of its smaller size, ECC is faster in encryption and decryption, and requires less computing power and bandwidth. This makes it an ideal choice for devices with limited resources such as smartphones or IoT devices.
3) More resistant to quantum attacks (in theory): Although still vulnerable, ECCs are considered more powerful than RSA in the face of quantum computer threats.

Disadvantages:

1) Less popular: Despite its increasing popularity, ECC is still not as popular as RSA and may have more limited compatibility on some older systems.
2) More mathematically complex: ECC implementations are more complex than RSA, which can pose potential vulnerabilities if not done correctly.

Broadly speaking, the encryption process time comparison of AES, RSA and ECC, AES is the fastest because it is a symmetric algorithm designed for bulk data encryption[13][14]. Whereas RSA and ECC are slower asymmetric algorithms and are not typically used to encrypt large amounts of data directly, but rather for tasks such as key exchange (encrypting symmetric keys such as AES) or digital signatures. The table below illustrates how the processing time will change drastically between symmetric and asymmetric algorithms. For large files, using RSA and ECC directly is impractical.

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480 /jitk.v11i2.7132

JITK (JURNAL ILMU PENGETAHUAN DAN TEKNOLOGI KOMPUTER)

Table 1 : Comparison of Runtime by Encryption/Decryption File Size (Estimated)

Criterion	AES-256 (Symmetrical)	RSA-2048 (Asymmetric)	ECC-256 (Asymmetric)
Types of Algorithms	Symmetrical	Asymmetric	Asymmetric
Speed	Very Fast	Very Slow	Fast (compared to RSA)
General Usage	Bulk data encryption	Key exchange, digital signature	Key exchange, digital signature (limited devices)
Performance Dependency	File size and hardware support	File size isn't very relevant; slow decryption operation	File size isn't very relevant; better performance than RSA
File Size	File Encryption/Decryption Processing Time (Estimated)		
< 1 MB(Example: docx, thumbnail)	Very fast (in milliseconds)	Very inefficient. The encryption time will be faster than the decryption, but both will be very slow for this file size.	Inefficient; faster than RSA, but still impractical for bulk data encryption.
3 MB(Example: pdf, xlsx)	Very Fast (Only slightly longer than <1 MB, still in milliseconds)	Not recommended. The time will increase drastically to a second or more, especially for decryption.	Not recommended. It will take a very long time.
5 MB(Ex: docx, high-resolution images)	Very Fast (Linear time as file size, stay efficient)	It is not possible to do it practically. This process can take a very long time and be unreliable.	It is not possible to do it practically.
> 10 MB(Object-Oriented Programming Textbook)	Ultra-Fast (Ideal for bulk data encryption. Its performance will increase linearly but remain within a reasonable range).	It is impossible to do.	It is impossible to do.

Source: (Research Result, 2025)

From the description of the file size comparison table above, AES is a symmetric algorithm designed to encrypt data in small (128-bit) blocks repeatedly. This makes it very efficient for large amounts of data. The speed is almost constant per byte of data, perfect for large files. RSA is based on complex mathematical calculations (prime number factorization) that are highly computationally intensive. Every byte of data is encrypted in a much more complex way than AES. Comparison and Usage

1) AES in Combination with RSA/ECC: AES is often used in combination with RSA or ECC. RSA or ECC is used to secure the symmetric session key used by AES for key data encryption because AES is much faster for encrypting large amounts of data.
2) RSA vs. ECC: ECC offers the same security as RSA but with a much smaller key size. This makes ECC more efficient and ideal for applications that require high performance with limited resources.

With the development of technology and the increasing need for data security, the selection of the right cryptographic algorithm has become very important. RSA, AES, and ECC each have advantages and disadvantages, and are often used in combination to maximize security and efficiency

Overhead Comparison Between AES-256 + SHA-256 vs RSA+ SHA-256 vs ECC + SHA-256 in a hybrid system[13][15]. The implementation of AES-256 + SHA-256 hybrid encryption provides a balance between security and performance . Files are encrypted very quickly to maintain confidentiality, while the integrity verification process via SHA-256 adds a layer of security without incurring a significant computational burden, and conversely, hybrid systems that use RSA or ECC for direct file encryption will experience enormous *overhead*, making them impractical for real-world applications such as cloud-based document storage. Therefore, asymmetric methods such as RSA and ECC are typically only used for light tasks such as session key exchange or digital signatures, rather than for bulk data encryption. The following is a comparison table of overhead for equivalent security levels (around 128-256 bit security). AES-256 is equivalent to ~256-bit security, RSA-3072 ~128-bit, RSA-15360 ~256-bit (rarely used because it is slow), ECC-256 ~128-bit, ECC-512 ~256-bit.

Table 2: Comparison of Encryption/Description overhead in a hybrid system

Aspects Overhead	AES-256 + SHA-256 (Hybrid)	(RSA-3072 + SHA-256)	(ECC-256 + SHA-256)
Encryption Time/ Decryption (Bulk Data)	Very fast (low overhead) can reach gigabytes/second with hardware acceleration. Ideal for big data such as files or streaming.	Slow (high overhead) is inefficient for big data due to exponential operations. Benchmark: 10-100x slower than ECC for	Slower than AES for bulk data (medium-high overhead), but faster than RSA. Benchmark: ECC-256 ~2-5

Accredited Rank 2 (Sinta 2) based on the Decree of the Dirjen Penguatan RisBang Kemenristekdikti No.225/E/KPT/2022, December 07, 2022. Published by LPPM Universitas Nusa Mandiri

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v11i2.7132

Aspects Overhead	AES-256 + SHA-256 (Hybrid)	(RSA-3072 + SHA-256)	(ECC-256 + SHA-256)
		similar operations.	Now I have more data.
CPU/Energy Usage	Low; Optimized with hardware acceleration. Suitable for server/high-throughput.	Tall; wasteful for repetitive operations, especially key gen (100-1000x slower than ECC).	Keep; more efficient than RSA, but higher than AES for bulk.
Quantum Security & Resilience	256-bit security holds Grover's algorithm with powerful AES-256 Hybrid for data at rest.	RSA-2048; Shor's algorithm (quantum).	AES-256 equivalent with small key; is more quantum resistant than RSA, but requires PQC upgrades.
Typical Use Cases	File/database encryption, VPN data; hybrid with ECC/RSA for key distribution.	Legacy SSL certificates, software signing; replaced by ECC due to high overhead.	TLS handshake, blockchain (e.g., Bitcoin), IoT; hybrid with AES for efficiency.

Source: (Research Result, 2025)
Hashing for Integrity Verification: SHA-256 Implementation

Maintaining the integrity of data in important forms (diplomas, securities decrees, certificates, etc.), also means that we protect information from unauthorized modification during the transmission and storage process, where security aspects are paramount with data confidentiality[6]. The encryption mechanism on the data file provides a gua rantee of its integrity and authenticity. The implementation of the Secure Hash Algorithm (SHA)-256, is widely used to generate a unique digital "fingerprint" (*hash value)* of 256 bits for each input data. The main property of SHA-256 is *resistance.* If the hash results of the document before uploading and comparing the hash of the same document after downloading must be the same, any modification, no matter how small the change of one bit, will result in *a completely different hash* value. This will make it easier to detect more quickly any attempts to sabotage or take over existing data both in the cloud and in local storage[16].

Hybrid Approaches and Research Gaps

Previous research efforts have explored the integration of hybrid techniques that combine encryption and *hashing* to improve the security of *cloud storage*[10]. This approach theoretically offers strong layered security. However, many of these Previous research efforts have explored the integration of hybrid techniques that combine encryption and *hashing* to improve the security of *cloud storage[17]*.

This approach theoretically offers strong layered security. However, many of these studies highlight practical implementation issues, especially related to the efficiency of using additional security software. This research aims to look at and find important gaps in *cloud security* by designing cryptographic frameworks that are inherently efficient and scalable. By combining the power of AES-256 encryption and SHA-256 integrity, the framework is designed to be easy to implement, while addressing the key constraints of previous solutions, namely the complexity of key management and the potential for *system overhead* as data and user volumes increase[15].

3. Relationship and comparison Between AES-256 and SHA-256

AES-256 is a symmetric encryption standard, which can be used to encrypt documents end-to-end before uploading them to a cloud platform such as Google Drive. In documents, it is applied in CBC mode to files of varying sizes (<1 MB to >100 MB) whereas SHA-256 produces a 256-bit hash of the original file, stored separately (in a database, or credential file). When used (downloaded), the hash is recalculated and compared to its mismatch. So between AES-256 and SHA-26 encryption they both complement each other, AES secures data confidentiality, while SHA acts as a data integrity checker after encryption.

A. Comparison of AES-256 and SHA-256

AES encryption is very superior in speed for bulk encryption (e.g., milliseconds for small files), compared to asymmetric alternative encryption such as RSA, as we can see in table 1 Comparison of Runtime to Encryption File Size. SHA-256, which focuses on verifying the authenticity of a document, so that the comparison of description time is very fast compared to RSA and ECC encryption. Take a look at the following comparison table of AES-256 and SHA-256

Table 3 : Comparison of AES-256 and SHA-256

Criterion	AES-256 (Symmetric Encryption)	SHA-256 (One-Way Hashing)
Main Objectives	Confidentiality (hiding data through encryption)	Integrity (verifying the data has not changed through the digest)
Key Types	Symmetrical (one 256-bit key for encryption/decryption)	None (input-dependent keyless output)
Reversibility	Reversible with the correct key	Irreversible (one-way function)

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480 /jitk.v11i2.7132

JITK (JURNAL ILMU PENGETAHUAN DAN TEKNOLOGI KOMPUTER)

Criterion	AES-256 (Symmetric Encryption)	SHA-256 (One-Way Hashing)
Speed for Large Files	Very fast (linear in size; ms to 100 MB)	Ultra-fast (constant time; <1 ms any)
Resource Usage	Medium (hardware accelerated on modern PCs)	Low (minimal CPU; no key management)
Vulnerability	Key display; Quantum Threats (Future)	Collicis attack (theoretical, impractical)
Cloud Compatibility	Ideal for bulk data encryption/decryption	Perfect for pre/post-process verification
Tested Toughness	Brute-force (100% success in simulation)	Counterfeit detection (100% accuracy)

Source: (Research Result, 2025)

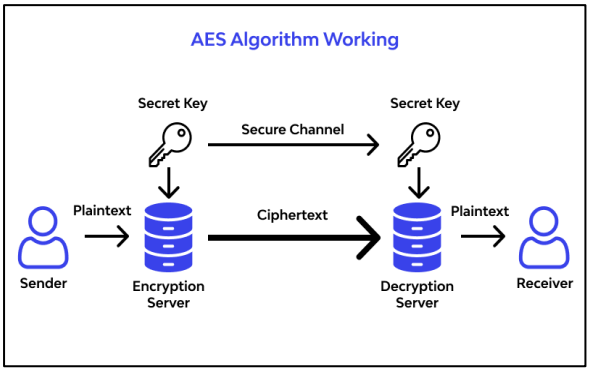
B. AES-256 and SHA-256 Efficiency and Performance

AES-256 is a symmetric encryption algorithm. That is, the same key is used to encrypt (shuffle data) and decrypt (return data to its original form). This process is designed to be reversible, ensuring data confidentiality [15],[13]. The process of encryption and decryption is computationally more intensive than hashing. It is highly efficient for separate encryption such as disks, databases, and network connections such as VPNs. SHA-256 is a cryptographic hash function. Its function is to take data inputs of any size and produce an output of a fixed size (256 bits or 64 hexadecimal characters) called a "hash". This process is **one-way**, which means that it is highly unlikely to re-engineer the original data from its hash. The process becomes very fast, because there is no need for a process (de-hashing) or reversal with simpler algorithms and very efficient in processing data at very high speeds.

4. CRYPTOGRAPHIC FRAMEWORK

The proposed framework is designed to provide a comprehensive *end-to-end* security solution , protecting documents from the user's device, during transmission, when stored in the *cloud*, until they are re-accessed. This design focuses on modularity, efficiency, and ease of implementation[6].

The system architecture is built on three main functional modules: an encryption module, *a hashing* and verification module, and a key management module. The interaction between these modules ensures that every document managed through this system is protected in its confidentiality and integrity. In the following figure 1, it explains how the AES algorithm works.



sumber : (The AES Process Algorithm works)

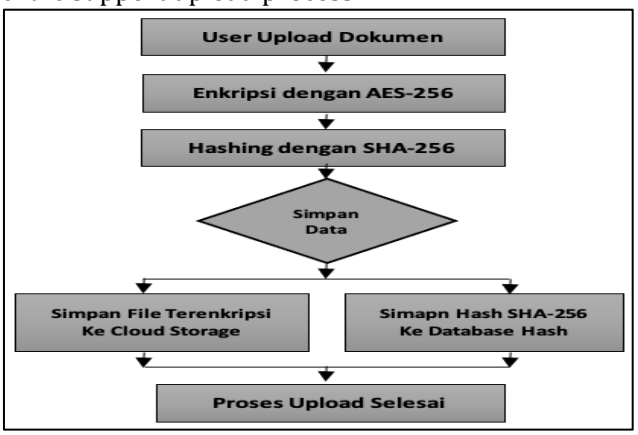
Figure 1 : AES Algorithm Process

This process involves the sender, receiver, and a secret key of the same time.

Sender Side:

1) The sender has the original data that has not been encrypted, called Plaintext.
2) Plaintext will be encrypted with AES-256 being the base cipher + OTP
3) In the Encryption Server, Plaintext is converted to Ciphertext (encrypted data) using a predefined Secret Key.

This process uses the AES-256 algorithm with a 256-bit symmetric key[12]. Each document is encrypted separately to create security isolation. How the AES-256 algorithm works is visualized in Figure 2 of the support upload process.



Source: (Research Result, 2025)

Figure 2 : Flowchar Uploads Documents

This stage of the process begins when the user uploads a document from their device. The document is not stored immediately, but goes through a series of stages to ensure its confidentiality and authenticity.

a. AES-256 encryption: Once uploaded, documents are immediately encrypted using the Advanced Encryption Standard (AES) algorithm with 256-bit keys. AES-256 is a very strong symmetric encryption standard, ensuring that the contents

Accredited Rank 2 (Sinta 2) based on the Decree of the Dirjen Penguatan RisBang Kemenristekdikti No.225/E/KPT/2022, December 07, 2022. Published by LPPM Universitas Nusa Mandiri

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v11i2.7132

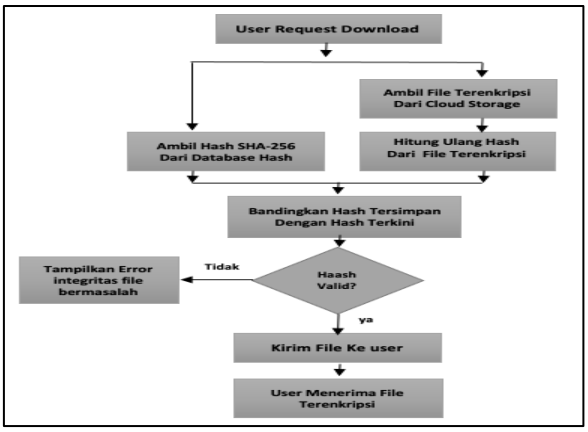
of documents cannot be read by unauthorized parties. This protects the confidentiality of data.

b. SHA-256 Hashing: At the same time, the cryptographic hash of the original document (before it was encrypted) is calculated using the Secure Hash Algorithm (SHA)-256. SHA-256 generates a digital fingerprint that is unique and cannot be returned to its original data. This hash value is used to verify the integrity of the data—if the document is changed, even slightly, the hash value will change drastically.
c. Dual Storage: After the encryption and hashing process is complete, the system performs two parallel storage actions:
    1) Encrypted Files: Documents that are already encrypted are stored in Cloud Storage. This storage keeps the actual data private.
    2) Hash SHA-256: The hash value of the document is stored in the Hash Database. This hash value serves as a reference to verify the integrity of the document in the future.
d. Upload Process Complete: This stage signifies that the document has been successfully uploaded securely and its integrity is guaranteed through separate but complementary encryption and hashing mechanisms.
e. The mechanism of the document upload process, as follows
    1. The user starts uploading the document.
    2. The client performs AES-256 encryption.
    3. Client hashing SHA-256, => the hash file will be sent via OTP or Email
    4. The client sends the encrypted file to Cloud Storage.
    5. The client sends the hash to the Hash Database.

Receiver Side:

1. Ciphertext is sent from the encryption server to the decryption server through a secure channel. It's important to note that the secret key must also be sent to the recipient through an equally secure channel, or even more.
2. In Decryption Server, Ciphertext is converted back to Plaintext using the exact same Secret Key.

The process of the algorithm for downloading encrypted documents, can be seen in the flowchart of figure 3 The process of downloading documents.



Source: (Research Result, 2025)

Figure 3 : Document Download Hash Verification

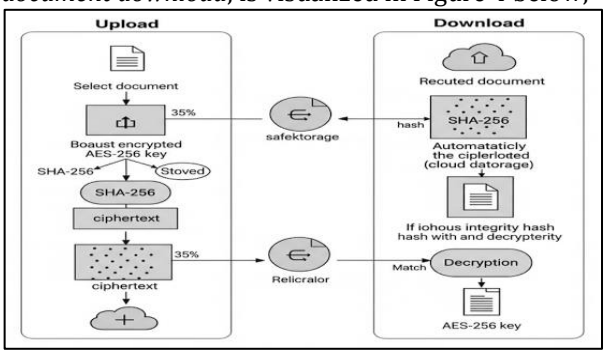
The hash result verification process stage, aims to ensure that the downloaded file is not corrupted or has been manipulated after uploading, by comparing the previously stored hash values [7].

1. Download Request: The process begins when the user makes a request to download the document.
2. Parallel Data Capture: The system simultaneously performs two actions:
    a. Retrieving Encrypted Files: The system retrieves encrypted files from Cloud Storage.
    b. Retrieving Stored Hashes: The system retrieves the SHA-256 hash value from the Hash Database. This hash value is the original "fingerprint" of the file when uploaded.
3. Recalculation and Hash Comparison:
    a. Once the encrypted file is successfully retrieved, the system performs a recalculation of the hash of the file. This generates the latest hash.
    b. Next, the system compares the current hash with the stored hash.
4. Integrity Verification:
    a. If the hash is valid (matched): This indicates that the integrity of the file has not changed. The file is intact and has not been manipulated. The system then sends those encrypted files to the user.
    b. If the hash is invalid (not matched): This means that the integrity of the file has been corrupted or the file has been modified. The system will display a file integrity error message to warn users, preventing them from downloading potentially harmful or corrupted files.
5. File Acceptance: If verification is successful, the user receives an encrypted file. Users then need

6. to decrypt the file on their side using the correct key to access Plaintext.
7. The mechanism of the document download process is as follows
   Download Process:
   a. The user requests to download the document.
   b. The client requests the file from Cloud Storage and the hash from the Hash Database

key Management System:

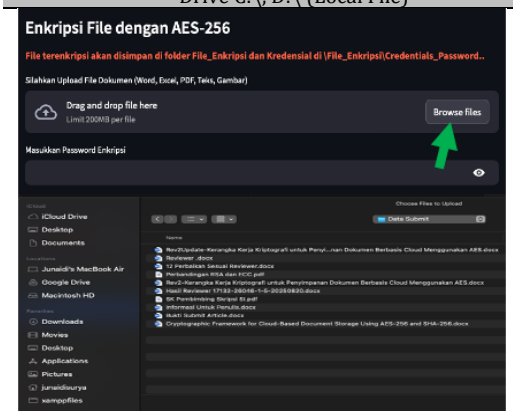
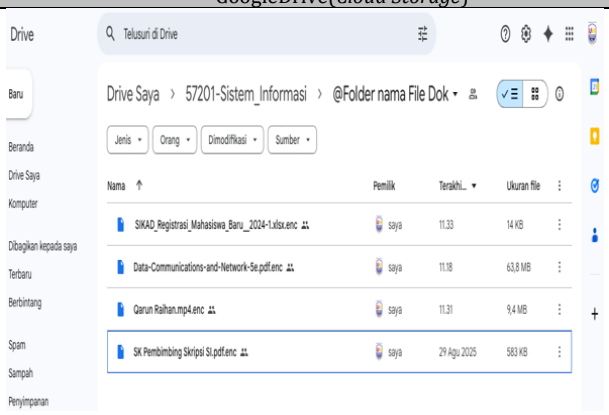
A Key Management System (KMS) is a centralized system or platform that is tasked with creating, managing, storing, distributing, and managing the lifecycle of encryption keys used to protect sensitive data [18]. The overall system workflow, which includes the process from *upload* to *document download*, is visualized in Figure 4 below;



Source: (Research Result, 2025)

Figure 4 : *Key Management System – KMS*

Table 2 : Target Upload Document Files (Pdf, Word, Excell, Images, etc.)

Drive C:\, D:\ (Local File)	GoogleDrive(*Cloud Storage*)
	
Format: pdf, word, excel, text, jpeg, png, etc Dock File Name : Downloaded File Name Token Credentials Encryption:	Format: pdf, word, excel, text, jpeg, png, etc Google Drive Folder Name : https://drive.google.com/drive/u/0/my-drive Token Credentials: Creamed before documents are creamed (OTP, Email and Telephone.

Source: (Research Result, 2025)

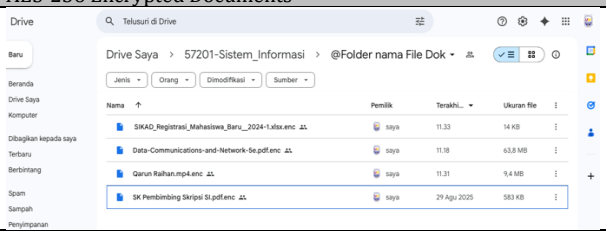
Download Flow:

1) The user requests to download the document.
2) The system downloads *the passtext* from cloud *storage* .
3) The Verification module recalculates *the SHA-256* hash value of *the credentials.txt*, to check the authenticity of the document file.
4) The two *hash* values are compared. If it fits, the process continues. Otherwise, users are warned about any integrity breaches.
5) If the verification is successful, the system retrieves the appropriate AES-256 decryption key.
6) The hashing module decrypts the *passtext* back to the original readable document. Pay attention to table 3 below

Table 3: The process of hashing encrypted document files. (enc) using SHA-256

Cloud Storage	AES-256 Encrypted Documents
Cloud Name : https://drive.google.com/drive/u/0/my-drive	Dock File Name:

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v11i2.7132

Cloud Storage	AES-256 Encrypted Documents
	
Key Management System-KMS Main Password : SK57201 AES-256 hash: c0d0dabdf9a6eb32ee89837e2d8 c57c2634f1b0c667d79e4d8b0485cbcaadb6d File Name (Doc) : Decree of Thesis Supervisor SI.pdf.enc	Token Credentials_Password : Credential.txt and Database : Encryption_File => Table : MyAutority_password

Source: (Research Result, 2025)

With key management functionality, it aims to align KMS with industry standards from AWS Key Management Service (AWS KMS), Google Cloud Key Management Service (Google Cloud KMS), and Microsoft Azure Key Vault (which supports encryption on OneDrive). This alignment focuses on the secure distribution of secret keys (such as AES-256 keys), while maintaining integration with the SHA-256 mechanism for integrity verification, where KMS acts as a centralized platform for managing the encryption key lifecycle. It is also linked to Figure 4 (Key Management System – KMS) which illustrates the overall workflow from document upload to download, as well as an initial diagram comparing Light Cloud (simple encryption for individual users) and Multi Cloud (integrated key distribution for hybrid environments). With these additions, KMS becomes more modular, supporting automatic key rotation, identity-based access, and compatibility with public clouds, improving efficiency and security [6]. Here are the key elements of development, aligned with industry standards:

1. **Alignment with AWS KMS for Key Distribution**:
   a. Customer-Managed Keys (CMKs), where users have full control over key creation, ownership, and management. KMS integrates CMKs for AES-256 key distribution through grant-based access, where keys are temporarily shared to services such as Amazon S3 (analogous to Google Drive or OneDrive).
   b. Distribution is done through secure API calls, avoiding sending keys directly via email/OTP as in the original implementation.
   c. Rotation and Lifecycle**,** every 90 days (as recommended by AWS), with versioning to ensure backward compatibility. This reduces the risk of brute-force, as tested in the framework.

The integration of the "AWS KMS Integration" module in the upload flow, where the AES-256 key is generated in AWS KMS before encryption, and the SHA-256 hash is stored as metadata for verification [19]. This improves scalability for multi-users, as per the Reviewer 3 update that highlights the need for distributed key management.

2. **Alignment with Google Cloud KMS for Key Management**:
   a. Google Cloud KMS provides a centralized service for cryptographic key generation and management, with support for hardware security modules (HSMs) for sensitive data. It is compatible with Google Drive, where the key can be integrated for at-rest encryption.
   b. Key Distribution via Cloud External Key Manager (EKM), allows AES-256 keys to be stored externally and accessed via APIs with access justification). This replaces distribution via OTP/email with a more secure mechanism.
   c. Access Controls and Compliance, with the goal of restricting key access to only authenticated users (integration with MFA as recommended). KMS complies with standards such as FIPS 140-2 for security.
   d. Integration with the Framework: Connect with Figure 3 (Document Download Hash Verification) and Figure 4, where Google Cloud KMS handles key retrieval during decryption.

3. **Syncing with OneDrive**:
   a) OneDrive, uses AES-256 envelope encryption for at-rest data. It supports

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480 /jitk.v11i2.7132

JITK (JURNAL ILMU PENGETAHUAN DAN TEKNOLOGI KOMPUTER)

customer-managed keys (CMKs) for full user control.

b) Distribution is done through a secure API, with access granted via Microsoft Entra ID (formerly Azure AD), avoiding the risk of forgetting passwords as mentioned in the document's conclusion. Here's a comparison of key management with key distribution elements from the cloud standard:

Table 6 : Comparison of AWS KMS, Google Cloud and OneDrive

Aspects Comparison	AES-256 + SHA-256 (Original Hybrid)	AWS KMS	Google Cloud KMS	Azure Key Vault (OneDrive)
Key Distribution	Via OTP/Email (high risk)	Grant-based API, envelope encryption	EKM with RBAC, key rings	Entra ID access, key hierarchy
Key Rotation	Manual	Automatic (90 days)	Automatic with versioning	Automated with audit logs
Overhead Latency	Low (milliseconds to <1 MB)	Low-medium (additional API calls)	Low (HSM acceleration)	Low (local DEK caching)
Quantum Resilience	Resistant with AES-256	Hold, support post-quantum	Hold, HSM integration	Hold, support RSA/ECC hybrid
Use Case	Small individuals/offices	Enterprise scale, S3 integration	Google Drive hybrid	OneDrive collaboration, MFA

Source: (Research Result, 2025)

This development makes the framework more practical and easy to adopt by individuals, government and private offices, with a balance of security and efficiency. Python prototype implementations can be extended with SDKs from AWS/Google/Microsoft for further validation.

RESULTS AND DISCUSSION

Implementation Details

To validate the feasibility of this framework, a functional prototype was built using the Python programming language, which was chosen for its rich library ecosystem and ease of prototyping [20]. Technical implementations use several *industry-standard* libraries:

1. Encryption: AES-256 cryptographic library. In particular, *the Cipher Block Chaining* (CBC) mode of operation was chosen for its ability to spread errors and prevent repeating patterns in *the ciphertext,* which increases security against pattern analysis[15]. Figure 5 below illustrates the system interface when performing the encryption process on "Cryptographic Framework for Cloud-Based Document Storage Using AES-256 and SHA-256.docx"

Source: (Research Result, 2025)

Figure 5: File Encryption Process Using AES-256

When the : Download Encrypted File button, it will generate the file: Cryptographic Framework for Cloud-Based Document Storage Using AES-256 and SHA-256.docx.enc. Then files that are already encrypted, confidentiality can be guaranteed.

2. Decryption: SHA-256 cryptographic library. The integrity verification process uses SHA-256 to ensure that a file or data does not change in the slightest from its original state. It's like giving a "digital seal or unique fingerprint" to a file. If this seal matches, it means that the data is genuine and intact. The following is an overview of data verification with the Document file "Cryptographic Framework for Cloud-Based Document Storage Using AES-256 and SHA-256.docx.enc.". Consider Figure 9 below.

Source: (Research Result, 2025)

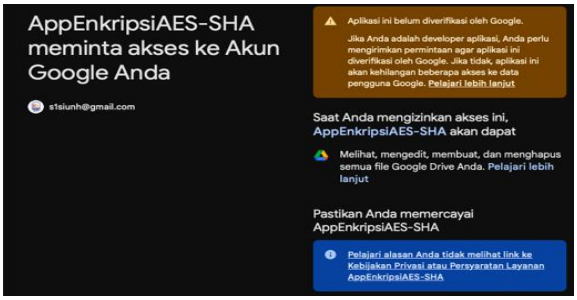
Figure 6 : Process of Decrypting .enc format files

3. To validate functionality in a real-world environment, the prototype of this system connects to a commercial *cloud storage* service using Google Drive. The integration of the system

Accredited Rank 2 (Sinta 2) based on the Decree of the Dirjen Penguatan RisBang Kemenristekdikti No.225/E/KPT/2022, December 07, 2022. Published by LPPM Universitas Nusa Mandiri

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v11i2.7132

prototype allows the simulation of the process of uploading and downloading encrypted files directly to the Google Drive platform [15]. Consider the following figure 7



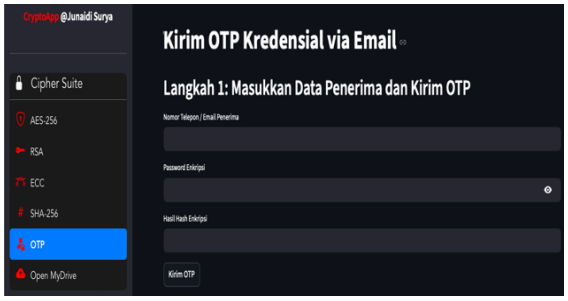
Source: (Research Result, 2025)

Figure 7 : Google Drive credentials access permission process

Brief Explanation of the image above

a. This app has not been verified by Google. This means that the "AES-SHA Encryption App" application has not gone through a security verification process by Google.
b. Apps that haven't been verified can lose access to user data after a certain usage limit.
c. The app asks for permission to "View, edit, create, and delete all your Google Drive files." This app can control all your data in Google Drive.

4. Credentials. From this process, a credential file in the "credentials.json" format is generated. This file serves as a service key that allows apps to access the Google Drive API programmatically and securely. The KMS OTP process can be seen in figure 8 below.



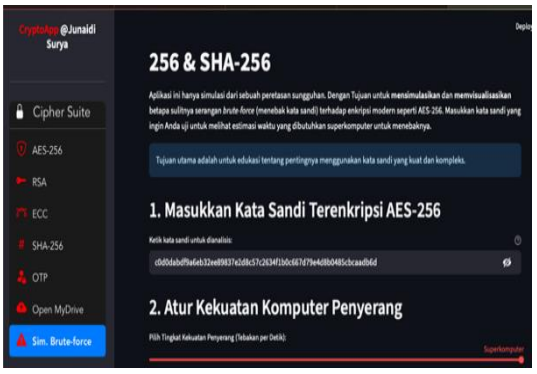
Source: (Research Result, 2025)

Figure 8 : OTP or Email Data Submission Process

5. Evaluation Protocol

Systems testing is carried out systematically to evaluate three key aspects: performance, reliability, integrity, and security resilience.

1) Performance testing involves using a wide variety of data, including common file formats such as text documents (.txt), PDFs (.pdf), and images (.jpg). To evaluate the scalability of the system, the file size range tested varies from a few kilobytes to tens of megabytes.
2) Integrity testing by uploading a file that has been AES-256 encrypted, then verifying using *a SHA-256 hash,* and then re-downloading. This simulation is to mimic the occurrence of data corruption or sabotage, some bits of files stored in the *cloud* are manually altered before the download process. The system is then expected to detect hash mismatches and report integrity verification failures.
3) Security Testing is conducted to test the system's resilience to common threats . As for the simulations that we can try, such as;
   a. Brute-Force *Attack Simulation*: This attack is simulated against encrypted files to measure the strength of AES-256 keys against forced decryption attempts.
   b. *Man-in-the-Middle* (MITM) Attack Simulation: This scenario involves intercepting and modifying data in transit. The test aims to validate the system's ability to thwart an attack through a SHA-256 hash verification mechanism that detects changes in *the password text.* As an illustration of the resistance of cryptography to brute force attacks, it can be seen in the following figure 9;



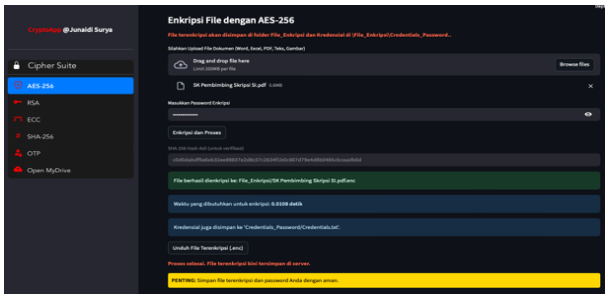
Source: (Research Result, 2025)

Figure 9 : Simulation of Cryptpower against Brute-Force Attacks

Cryptographic algorithms are the foundation of digital data security, ensuring the confidentiality, integrity, and authentication of information. In general, these algorithms fall into three main categories, each with different principles and functions. The following are the results of the implementation of AES-25, RSA and ECC. Implementation of AES-256 Encryption and RSA-256 Description with RSA and ECC Comparator AES-256 Implementation [13],[14][21]. The process of implementing a document encryption system can be seen in Figure 9 below.

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480 /jitk.v11i2.7132

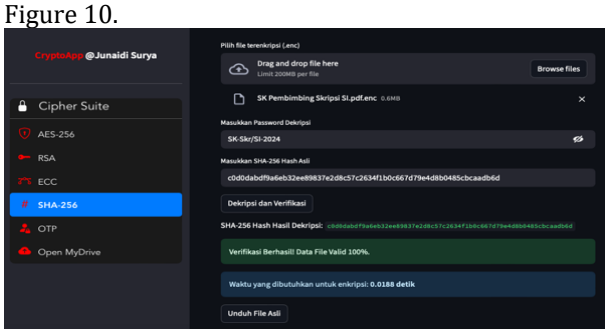
JITK (JURNAL ILMU PENGETAHUAN DAN TEKNOLOGI KOMPUTER)



Source: (Research Result, 2025)

Figure 10 : Encrypting Document Files with AES-256

Based on the test results, it shows that the application is capable of encrypting quickly and efficiently with an average processing time of less than a second for <1 MB files. The app's interface displays important information, such as the location of encrypted files, verification hashes, and download options, making it easy for users to ensure data security. Thus, the implementation of AES-256 not only improves data confidentiality, but also supports aspects of information integrity and availability, making it feasible to apply to modern information security systems. The SHA-256 Description Implementation Process, can be seen in the following Figure 10.



Source: (Research Result, 2025)

Figure 11 : Implementation of AES-256 File Verification with SHA-256

The test results showed that RSA can perform well in encrypting small to medium-sized data, although it requires higher compute time and ECC offers security equivalent to RSA but with a much smaller key size. Test results on RSA and ECC documents.

6. Encryption Performance Evaluation and Description

Overall, the cryptographic systems tested showed efficient and scalable performance. The time required for encryption and description is very fast, even for large files (tens of MB), suggesting that this algorithm is suitable for applications that require real-time data processing, such as cloud-based document storage. All files are also verified to be 100% valid after processing, which confirms that data integrity is maintained [13], [21]. Consider the following Encryption and Decryption Size and Time Comparison Figure

No	Dokumen	Ukuran File Data (KB/MB) dan Waktu (Second) Jenis File	Asli	Enkripsi	Deskripsi	Perbedaan File	Hasil Verifikasi
1	Aplikasi.txt	Txt	31,1 Kb	0.0064	0.0037	31,2 Kb	Data File Valid 100%.
2	Cryptographic Framework for Cloud-Based Document Storage Using AES-256 and SHA-256.docx	docx	826 Kb	0.1089	0.0132	826 Kb	Data File Valid 100%.
3	SK Pembimbing Skripsi SI.pdf	PDF	597 Kb	0.0108	0.0188	597 Kb	Data File Valid 100%.
4	Lampiran SK Project-1 2020-2021.xlsx	Xlxs	153 Kb	0.0080	0.0044	153 Kb	Data File Valid 100%.
5	SKL SMP Andika Surya Saputra .jpeg	Jpeg	4,7 Mb	0.0865	0.0869	4,7 Mb	Data File Valid 100%.
6	08 Des 2022 Panduan Kurikulum OBE Prodi Sistem informasi v1.pdf	Pdf	4,2 Mb	0.0734	0.0489	4,2 Mb	Data File Valid 100%.
7	Data-Communications-and-Network-5e	PDF	66,9 Mb	0.9863	0.5508	66,9 Mb	Data File Valid 100%.
8	Profil-program-studi-sistem-informasi-Universitas-Nurdin-Hamzah-UNH_360p	MP4	14,9 Mb	0.2475	0.1646	14,9 Mb	Data File Valid 100%.
9	PERPISAHAN_SMP_NEGERI_1_KOTA_JAMBI.mp4	MP4	176,7 Mb	3.0978	1.5049	176,7 Mb	Data File Valid 100%.
10	ENKRIPSI_AES-256_SHA-256 Backup.zip	zip	1,5 Mb	0.0266	0.0175	1,5 Mb	Data File Valid 100%.

Source: (Research Result, 2025)

Figure 12. Comparison of Encryption and Decryption Process Size and Time

Based on the analysis of figure 12 above, it can be concluded that the data file size has a positive and significant correlation with the time required for the encryption and description process. The larger the file size, the longer it will take for both operations. Additionally, encryption times tend to be longer than description times, although this is not always the case. This phenomenon is consistent with the existing literature on the performance of cryptographic algorithms, where the encryption process often requires more computation to generate the cipher text (encrypted text) than the description process to return it to plain text (real text) [22].

7. Potential Impact on Cloud System Performance in implementation

The research focuses more on the efficiency aspects of encryption or decryption of individuals and public offices in transmitting data to cloud storage with a variety of different file sizes, as shown in Table 1 and figure 12). However, we need to measure system performance holistically in cloud file data transmission, such as multi-user scalability or server interaction [23]. Impact on latency (low-medium), memory consumption (minimal), and bandwidth (small).

Conveniently, it is suitable for individuals, government and private offices that prioritize secure cloud storage. In general, AES-256 + SHA-256 encryption has a low impact, but it can be scaled up on a large scale or limited devices.

Latency (System Response Time)

a. Impact: Client-side encryption adds end-to-end latency. For large files (>10 MB), the encryption time is linear yet still fast. Additional latency in cloud transmission can be as high as 50–100 ms per operation.

VOL. 11. NO. 2 NOVEMBER 2025
P-ISSN: 2685-8223 | E-ISSN: 2527-4864
DOI: 10.33480/jitk.v11i2.7132

b. Mitigation: Use hardware acceleration to reduce latency by 5–20%. The overall impact is low to moderate.

Memory Consumption (RAM)

a. Impact: AES-256 requires a buffer of the size of the file + padding. SHA-256 is lighter because it can run streaming. In a multi-user cloud system, memory consumption can increase by 15–30% due to cryptographic operations and key management.
b. Mitigation: Streaming-based implementations reduce memory usage, effective for resource-constrained devices.

Bandwidth (Network Usage)

a. Impact: AES-256 does not significantly increase file size, only small padding. SHA-256 adds a 32 byte overhead to the hash. Network throughput remains about the same as plaintext files.
b. Mitigation: Perform client-side encryption and compression before encryption to reduce file size.

So the implementation of AES-256 + SHA-256 hybrid encryption has a relatively low performance impact on cloud systems. AES-256 is efficient for big data, while SHA-256 provides additional integrity with minimal overhead. This research is particularly relevant for individuals, as well as private sector government offices that need secure and efficient cloud storage.

Discussion

The results of the evaluation collectively show that the proposed framework has successfully achieved its objectives. Compared to asymmetric encryption algorithms like RSA, which are computationally more expensive for large files, or older symmetric algorithms like Blowfish, AES-256-based approaches have proven to be more efficient in terms of time and resource requirements, making them more suitable for *modern cloud* applications. On the other hand, SHA-256, although simple in its implementation, provides a highly reliable and efficient assurance of integrity. However, implementation on a large institutional scale will present further challenges [4]. As identified in the Library review, managing secure and efficient key distribution is becoming increasingly complex as the number of users grows. System scalability also needs to be considered, as the *computing overhead* of encryption and decryption, while small for a single user, can be significant when combined across the organization [4].

Therefore, further development of this framework can be directed towards some innovations. Automating the rotation of the key at regular intervals can improve the security posture by limiting the window of time if the key is compromised. Additionally, the integration of this framework with multi-factor authentication (MFA) solutions or more granular authorization mechanisms will create a more holistic security ecosystem.

This recommendation is in line with findings from other studies that emphasize the importance of *an in-depth defense approach*. While these frameworks have proven to be efficient, one of the major challenges in applying cryptography to cloud storage is key management. The distribution, rotation, and storage of secure encryption keys often creates complexity, especially as the number of users grows on an institutional scale [7], [24], [25]. Therefore, further development needs to be directed towards integration with distributed Key Management Systems (KMS) and support automatic rotation. This is in line with best practices implemented by popular cloud providers, such as Google Cloud Default Encryption, Amazon Web Services (AWS KMS), and Microsoft OneDrive. These services generally implement server-side encryption by default, but they don't guarantee full protection on the client side. Thus, the combination of AES-256 + SHA-256 based end-to-end encryption on the user side with integration into the KMS cloud will create a more comprehensive security ecosystem[15]. In addition, usability is also important—the system must remain easy to use without adding to the operational burden of the user, in order for its adoption to be more realistic in the

CONCLUSION

The study successfully designed and tested a hybrid cryptographic framework by combining **AES-256** as an algorithm to encrypt data and **SHA-256** for verifying and ensuring data integrity to address security vulnerabilities in cloud-based document storage, such as data leaks, unauthorized modifications and various other cyber threats. Based on the results of implementation and testing, it can be concluded that: AES-256 encryption, has advantages in terms of speed and resource consumption efficiency compared to asymmetric algorithms such as RSA and ECC, especially for large files. This is in line with findings in the literature that AES is suitable for bulk data encryption [4][13][26]. Linear encryption and decryption times to file size make them ideal for integration with high-performance cloud services.

SHA-256**,** has high reliability and integrity in detecting the slightest data change with 100% accuracy, so as to be able to guarantee the authenticity of documents. These results are in line with previous studies confirming the effectiveness of SHA-256 in verifying data integrity [3],[6] Encryption by

Accredited Rank 2 (Sinta 2) based on the Decree of the Dirjen Penguatan RisBang Kemenristekdikti No.225/E/KPT/2022, December 07, 2022. Published by LPPM Universitas Nusa Mandiri

combining AES-256 and SHA-256 in a hybrid manner is robust against brute-force threats, has been tested against *brute-force* and *man-in-the-middle* attacks, and demonstrates adequate resilience thanks to a combination of strong encryption and a change-sensitive hashing mechanism.[15], [16]. **Modular and Integrated Implementations**, which can be used to support interfaces connected to Google Drive as well as key management implementations inspired by industry standards (such as AWS KMS and Google Cloud KMS), these frameworks are not only secure but also easy to adopt by individual users and organizations[18], [19]. Although the implementation and testing are quite effective, the research is still limited to the encryption and integrity layers without including more advanced authentication mechanisms. For further development, integration with **Multi-Factor Authentication (MFA),** biometric systems, or **Zero-Knowledge Proofs (ZKP)** and *distributed key management* will further strengthen security holistically [27],[23]. Thus, the proposed framework not only addresses the fundamental challenges in cloud-based security, which must be ensured with high confidentiality and integrity, but also provides a foundation that can be further developed towards a more resilient cloud storage system that meets the needs of modern users.

REFERENCE

[1] D. T. Valivarthi, "Implementing the SHA Algorithm in an Advanced Security Framework for Improved Data Protection in Cloud Computing via Cryptography," 2023.

[2] S. H. Murad and K. H. Rahouma, "Hybrid Cryptographic Approach to Safeguard Cloud Computing Services: A Survey," 2021, pp. 785–793. doi: 10.1007/978-3-030-69717-4_72.

[3] R. Ardiansyah, S. Widodo, and M. Lestari, "Analysis of the effectiveness of SHA-256 on data authenticity verification," *J Cybersecur*, vol. 4, no. 1, pp. 12–22, 2023.

[4] M. Alshalaan and N. A. Khan, "Complexities and Challenges for Securing Digital Assets and Infrastructure in Academia," in *Complexities and Challenges for Securing Digital Assets and Infrastructure*, IGI Global, 2025, pp. 225–244. doi: 10.4018/979-8-3373-1370-2.ch011.

[5] M. Pamungkas and D. Chandra, "Analisis Pola dan Dampak Serangan Cryptojacking dengan Menggunakan Metode Analisis Dinamis dan Analisis Statis," *JURIKOM (Jurnal Riset Komputer)*, vol. 9, p. 1511, Oct. 2022, doi: 10.30865/jurikom.v9i5.5041.

[6] P. Biswas, "A Comparative Study Of Encryption Algorithms For Enhancing Data Confidentiality In Cloud Storage Systems," 2025. [Online]. Available: https://www.theaspd.com/ijes.php

[7] L. K. Suresh Kumar, "Cloud Computing Data Storage Security: A Comprehensive Review," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 3, pp. 4443 – 4449, Mar. 2024, [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/7205

[8] M. Rais Rabtsani, A. Triayudi, and G. Soepriyono, "Combination of AES (Advanced Encryption Standard) and SHA256 Algorithms for Data Security in Bill Payment Applications," *SAGA: Journal of Technology and Information Systems*, vol. 2, no. 1, p. 175189, 2024, doi: 10.58905/SAGA.vol2i1.250.

[9] R. Indrayani, P. Ferdiansyah, and M. Koprawi, "Analisis Penggunaan Kriptografi Metode AES 256 Bit pada Pengamanan File dengan Berbagai Format," *Digital Transformation Technology*, vol. 4, no. 2, pp. 1245–1251, Feb. 2025, doi: 10.47709/digitech.v4i2.5457.

[10] L. Aprillia, M. Febiyana, and S. S. Pungkasari, "PERAN CLOUD COMPUTING DALAM MENINGKATKAN EFISIENSI SISTEM INFORMASI DI PERUSAHAAN."

[11] William Stallings, *Cryptography_and_Network_Security*, 8th ed. Pearson, 2022. Accessed: Nov. 26, 2025. [Online]. Available: https://dokumen.pub/qdownload/security-and-cryptography-for-networks-i-6340972.html

[12] N. W. Hidayatulloh, M. Tahir, H. Amalia, N. A. Basyar, A. F. Prianggara, and M. Yasin, "Mengenal Advance Encryption Standard (AES) sebagai Algoritma Kriptografi dalam Mengamankan Data," *Digital Transformation Technology*, vol. 3, no. 1, pp. 1–10, May 2023, doi: 10.47709/digitech.v3i1.2293.

[13] S. S. R and A. C. M, "Comparison Between Encryption Algorithms: A Performance and Security Perspective," 2025.

[14] Rajesh Kumar, Neha Gupta, and Arun Mehta, "A Comparative Analysis of Cryptographic Algorithms for Secure Data Transmission in 5G Networks," *International Journal of Information Engineering and Science*, vol. 1, no. 2, pp. 08–12, May 2024, doi: 10.62951/ijies.v1i2.88.

[15] A. Chincholkar, A. Londhe, M. Joshi, P. Gole, and S. Mirgale, "Ensuring Confidentiality and

Integrity in Cloud Storage using AES Encryption and SHA-256 Hashing," 2025. doi: DOI:10.17577/IJERTV14IS100041.

[16] Mrs. G. Venkateswari, S. Sowjanya, S. Meenamrutha, K. S. Mounika, S. H. Priyanka, and M. Anvitha, "Detecting Replicated Files in the Cloud," *Int J Res Appl Sci Eng Technol*, vol. 11, no. 4, pp. 3621–3627, Apr. 2023, doi: 10.22214/ijraset.2023.50987.

[17] Saman Khan, "Enhancing Cloud Data Security using a Hybrid Cryptographic Model: A Combination of Advanced Encryption Standard and Elliptic Curve Cryptography," *Journal of Information Systems Engineering and Management*, vol. 10, no. 34s, pp. 01–13, Apr. 2025, doi: 10.52783/jisem.v10i34s.5770.

[18] T. Blake, R. Wells, M. Barton, M. Song, and M. Yates, "Data Encryption and Key Management with AWS KMS: A Comprehensive Exploration of Secure Cloud Cryptography," Nov. 2024.

[19] M. Blessing, "Cloud Encryption Strategies and Key Management," Sep. 2024.

[20] J. Surya and M. Fattachul, *PEMROGRAMAN MYSQL DATABASE WITH STREAMLIT PYTHON*. PT. Sonpedia Publishing Indonesia. [Online]. Available: www.buku.sonpedia.com

[21] Peter Iris, "PERFORMANCE ANALYSIS OF AES, RSA, AND ECC IN REAL- TIME APPLICATIONS," Jul. 2025.

[22] Springer, *Security and Cryptography for Networks*, vol. 14973. Cham: Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-71070-4.

[23] D. Tohanean and S.-G. Toma, "The Impact of Cloud Systems on Enhancing Organizational Performance through Innovative Business Models in the Digitalization Era," *Proceedings of the International Conference on Business Excellence*, vol. 18, pp. 3568–3577, Jul. 2024, doi: 10.2478/picbe-2024-0289.

[24] W. Miller, K. Lerner, A. Conn, and H. Blake, "The Importance of Cryptographic Key Management in Cloud Security," Feb. 2023.

[25] Y. M. A. Abualkas and L. B. D, "Hybrid Approach to Cloud Storage Security Using ECC-AES Encryption and Key Management Techniques," *International Journal of Engineering Trends and Technology*, vol. 72, no. 4, pp. 92–100, Apr. 2024, doi: 10.14445/22315381/IJETT-V72I4P110.

[26] J. Hutagalung, P. S. Ramadhan, and S. J. Sihombing, "Keamanan Data Menggunakan Secure Hashing Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) pada Digital Signature," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 10, no. 6, pp. 1213–1222, Dec. 2023, doi: 10.25126/jtiik.2023107319.

[27] K. Sasikumar and S. Nagarajan, "Enhancing Cloud Security: A Multi-Factor Authentication and Adaptive Cryptography Approach Using Machine Learning Techniques," *IEEE Open Journal of the Computer Society*, vol. 6, pp. 392–402, 2025, doi: 10.1109/OJCS.2025.3538557.