International Journal of Computations, Information and Manufacturing (IJCIM) 4(2) -2024

Contents available at the publisher website: G A F T I M . C O M

International Journal of Computations, Information and Manufacturing (IJCIM)

Journal homepage: https://journals.gaftim.com/index.php/ijcim/index

Study on a Hybrid Encryption Method for Data Security Based on ARB3 Algorithm

Vijay Vamsi Nadakuditi[1], *Radhika Rani Chintala*[2]

[1,2]*Dept. of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, AP, India*

A R T I C L E I N F O

Keywords:
ARB3 Algorithm 1, Encryption Performance Optimization 2, Data Integrity 3, Hybrid Cryptography 4

Received: Aug, 18, 2024
Accepted: Oct, 28, 2024
Published: Dec, 25, 2024

A B S T R A C T

In a world where cloud storing and online correspondence will increase in a faster pace, it will be very important that the cryptographic safeguarding will be such that will be solid and powerful. Even where traditional hybrid techniques of encryption have been found to be effective, it will not be spared of the problem of throughput bottleneck, performance pitfalls, and security vulnerabilities. To ensure synergy we will solve these problems using ARB3, a new algorithm to implement a hybrid cryptography approach to merge the strengths of Rivest-Shamir-Adleman, BLAKE3 hashing, and Advanced Encryption Standard into a new algorithm ARB3. The ARB3 Algorithm will take advantage of the high-performance and symmetric encoding of the AES, the trustworthy key exchange process of RSA, and, high performance hashing of BLAKE 3 to develop an inclusive security protocol. Using testing, we shall have the advantage of showing that ARB3 will be better in terms of processing speed and provide an innocuous mix of security characteristics and that it will surpass traditional hybrid models in encryption and decryption. The general increment in throughput in ARB3 will allow it to receive large amounts of data without affecting its security or speeds. The proposed solutions will not only demonstrate how effective ARB3 will be in eliminating security threats, but they also will play a significant role in upgrading its performance metrics, which will become critical in future data security. Since the restrictions of the existing hybrid cryptographic systems will be reconsidered, ARB3 is going to provide a more promising and effective way of converting and processing data in the framework of many applications. It will be a major breakthrough in the cryptography sphere. The paper will also highlight the relevance of this new hybrid mechanism of encryption that will be fitted to the changing requirements of data security.

1. INTRODUCTION

Cryptography systems are important in assuring information security. Cryptography form's part and parcel of modern protection of information. Economic defines cryptography as the act of altering the messages that can be read to the ones that cannot be read in a manner that the genuine receiver of the message can read the message. Cryptograph is a word that starts with the Greek words kryptos, which means hidden or secret, and graphia, which means writing (Sharma et al., 2023). Cryptographic algorithms are very instrumental in maintaining confidentiality, authenticity, and integrity of data (Fernando et al., 2019). Cryptographic systems are commonly classified into symmetric, asymmetric and the hash- based cryptographic systems. The two of them possess their advantages and drawbacks:

1.1. Symmetric cryptography

Symmetric encryption makes use of a common key both in encryption and decryption. Many well-known symmetric algorithms are AES, DES, and Blowfish (Zhang, 2021). The most commonly used

AES is fast allowing a time complexity of about O(1) per block when used normally. Symmetric cryptography, however, is blind to the secure distribution of keys, and that becomes a big problem in an untrusted environment.

1.2 Asymmetric Cryptography

Public-key cryptography or more generally asymmetric cryptography, utilizes two different keys: an encryption key, which is called a public key, and a decryption key, which may be called a secret key or a private key (Meng & Wang, 2022). As an example, RSA offers good security features like authentication, non-repudiation and integrity (D. P. et al., 2020). But the operations of RSA are very computationally heavy, with a time complexity of $O(n^3)$ measured in terms of key size. This is not efficient in encrypting big volumes of data.

1.3 Hash Function

A hash function is a function that turns input content into a fixed-size digest and provides properties of determinism, efficiency, and resistance to pre-image and collision attacks (Jirwan et al., 2019). Algorithms that are now considered insecure, such as MD5, have been retired; however, SHA-256 is considered secure. Additionally, a more recent alternative, BLAKE3 makes hash functions considerably faster than classical hash functions (and runs at upwards of 4 GB/s on a modern CPU) and provides equivalent cryptographic assurances (Timothy & Santra, 2017).

1.4 Hybrid cryptography

Hybrid cryptography is the idea that combines several different cryptographic approaches and the ability to gain high-performance outcomes by lessening the compromise of speed versus security. A symmetric key is typically used to encrypt data which is transmitted securely with an asymmetric key. Hybrid cryptography is the technique for better overall compute costs and usage of a secure key exchange (Jintcharadze & Iavich, 2020).

Hybrid cryptography is currently being implemented much more widely as hybrid cryptography describes some of the following examples:

• HTTPS/SSL/TLS - based on the asymmetric key exchange, and a symmetric encryption of a session.

• E-mail Encryption (PGP/S-MIME) - having the symmetric keys are sad secured with an asymmetric methodology.

• Cloud Storage - provided many companies fast encryption and secured key (Kumar et al., 2020).

• VPN - used an RSA algorithm to exchange and the key to enter a tunnel and associate the tunnel with AES.

• E-commerce - secured highly sensitive information with bulk encryption and key management process through the use of hybrid cryptography.

1.5 Why Prefer Hybrid Cryptography

The combination of asymmetric and symmetric encryption enables hybrid models to outperform individual techniques in both speed and security (Hyseni et al., 2018). For example:

• AES can encrypt data at approximately 1.5 GB/s, whereas RSA is only efficient for small key sizes.

• RSA-2048 key operations may take several milliseconds per transaction, while AES-128 can process blocks in microseconds.

• Hybrid models minimize key management risks while enabling efficient large-scale data encryption.

1.2 Advanced Encryption Standard

The Advanced Encryption Standard is referred to as AES. (Lu & Tseng, 2002) The symmetric block cipher algorithm family includes this one. To encode and decode, it uses a key with 128, 192, or 256 bits. As seen in Figure 1. this technique makes use of an SP network with various rounds, which again rely on the size of the key being used. (Kumar et al., 2021). In every round, there are four stages:

1. In the initial phase, we replace the bytes.
2. In the second phase, we move the rows.
3. We then combine the columns.
4. After that, we give it the round key.

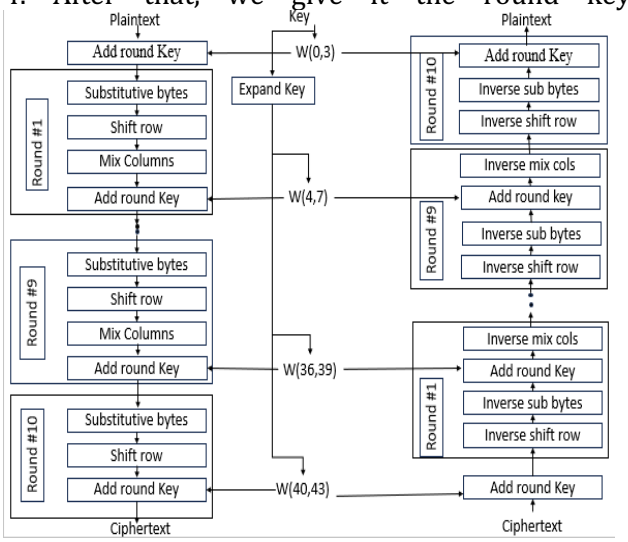

Figure 1: AES Encryption and decryption system

1.3 Rivest-Shamir-Adleman (RSA)

RSA is asymmetric, means it uses two distinct keys: a private key and a public key. The private key is kept private, while the public key is transfer with everyone. Rivest, Shamir, and Adleman are recognized for creating the RSA algorithm. The most well-known feature of RSA is its secure encryption method. The exponential usage of positive integer prime integers for encryption and decryption is the core idea behind RSA. (Aufa et al., 2018) The RSA algorithm operates as indicated in Figure 2.

Two exponent variables are used in this method: e for public and d for private. M and C are used as plaintext and ciphertext, respectively, to define the encryption and decryption procedure as follows: the encryption and decryption procedure (Ciocan et al., 2021).

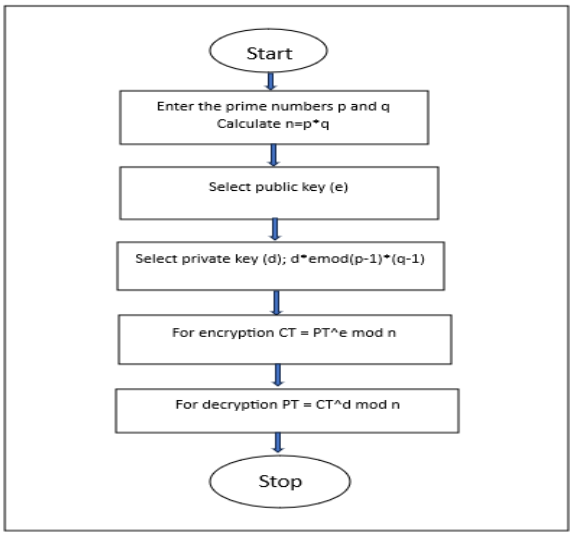

Figure 2. RSA Algorithm

1.4 Blake 3

It is a cryptographic hash function developed to offer integrity and authentication of data while having higher efficiency and security with a fixed-size hash output. The primary functionality of obtaining one-time hash from the input data in Blake3 is to map integrity in data. In this way, Blake3 carries much importance in holding accuracy along with trustworthiness of transmitted information.

The performance. Software comparing the two SHA-3 finalists' performance shows that Blake is almost three times faster than Keccak on a contemporary CPU for a 512-bit hash (Kahri et al., 2013), but only 1.26 times better for a 256-bit hash. Performance is expressed in cycles per byte (Ahmad & Garko, 2019).

The primary steps involved in Blake3's operation are as follows:

Chunking*:* The input text is separated into discrete pieces, usually with a size of 1 KB. Parallel execution is made possible by the independent processing of these portions. Padding is applied to finish the chunk if the data is less than 1 KB. Every chunk has attributes like chunk length and whether it is the last chunk, along with an identifier (such as its position in the data). (Baqtian & Al-Aidroos, 2023).

Compression Function*:* Compression Function: A compression function that carries out cryptographic mixing receives each chunk. This function applies operations like addition, XOR, and bit rotation on a set of 32-bit words. It is based on Blake2's permutation function. Simple patterns won't show up in the output since the mixing stage makes Sure the input data is evenly distributed throughout the chunk.

Merkle Tree Structure*:* A binary Merkle tree is built using the hash values from every chunk. The process proceeds up the tree until a single root node is created by combining chunks in pairs and compressing their hashes to create parent nodes. Because of this structure, Blake3 may effectively enable parallel computation by processing chunks separately before joining them (Sharma et al., 2019).

Finalization*:* Using particular domain settings and keys, the root node—which stands for the sum of the hashes of all the chunk goes through a last compression stage. This process guarantees the final hash's security and uniqueness. To reach the required length, usually 256 bits, which is the average output size for Blake3, the output is either truncated or expanded.

Output*:* The final hash is displayed in hexadecimal format as a 256-bit digest. This digest can be used as a fingerprint for data comparison or to confirm the integrity of the data, among other uses (Verma et al., 2021).

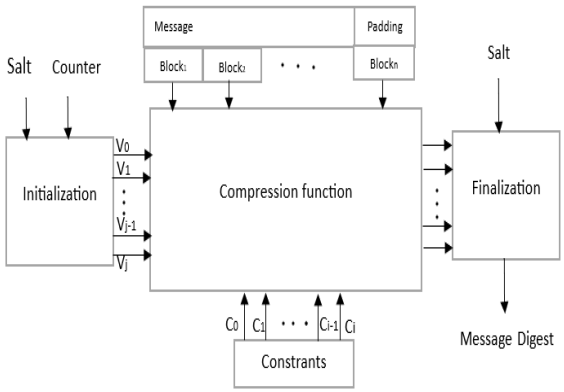

Figure 3: Architecture of the Blake3 hash function
As demonstrated in the architectural diagram of Blake3 Figure 3. a hash function accepts the input, and it gives the fixed-size hash value that uniquely represents the data. Therefore, it is very efficient and secure; hence it can be applied in hybrid systems for data integrity verification. Hybrid cryptography checks whether there is any change in the data during transmission or not. It does this by taking the hash values of the plaintext data before and after transmission and comparing them.

Table 1: The comparison table presents an analysis of various hybrid cryptographic algorithm

Combination	Enc/Dec Time (s)	Memory Usage (MB)	Time Complexity	Throughput (bytes/sec)	Cryptographic Strength
AES + DES + RSA [17]	16.93 / 11.71	20.51 / 16.28	AES:O(1), RSA: $O(n^3)$, DES: O(1)	16,936,330.00	AES-128, RSA-2048, DES-56 (vulnerable)
AES + RSA [18]	18.48 / 13.08	12.01 / 16.00	AES: O(1), RSA: $O(n^3)$	13,285,474.46	AES-128, RSA-2048
AES + RC6 + Blowfish [18]	17.00 / 27.00	14.08 / 12.89	AES: O(1), RC6: O(1), Blowfish: O(1)	9,532,509.00	All 128-bit symmetric keys; moderate resistance
RSA + MD5 [19]	30.69 / 30.25	10.35 / 16.74	RSA: $O(n^3)$, MD5: O(n)	6,881,289.00	RSA-2048, MD5 (128-bit; known collisions)
RSA + HMAC [20]	38.02 / 33.15	12.66 / 15.79	RSA: $O(n^3)$, HMAC: O(n)	5,892,200.49	RSA-2048, HMAC-SHA2 (256-bit; strong)
AES + MD5 [21]	18.93 / 14.65	12.73 / 16.09	AES: O(1), MD5: O(n)	12,486,131.99	AES-128, MD5 (128-bit; weak)
Blowfish + SRNN [22]	11.93 / 11.84	13.00 / 9.00	Blowfish: O(1), SRNN: varies	10,048,576.00	Blowfish-128, SRNN (uncertain; depends on model)

2.RELATED WORK

In order to improve a file's security features, this study combines file deduplication with Advanced Encryption Standard encryption. Deduplication was accomplished using the Blake3 hashing technique, which allows duplicate data blocks to be removed in order to maximize storage capacity. AES encryption ensures strong confidentiality of unique files. In this instance, it was striking a balance between robust encryption and data optimization. This strategy is better suited to the problems with cloud computing, where security and efficiency are the main requirements. (Chinnasamy & Deepalakshmi, 2018) The study examines current developments in these fields, emphasizing the importance of integrating encryption and deduplication to create a safe yet effective cloud architecture (International Journal of Mathematical Sciences and Computing, 2020).

The study focuses on a sophisticated encryption processing system that uses a modified DES algorithm to process accounting data. (Gajendra et al., 2016) By optimizing the system architecture from hardware acceleration points and integrating the more secure AES algorithm than the DES algorithm, the authors show significant time savings for encryption activities. This demonstrates that in addition to improved security training, appropriate key management, backup,

and recovery procedures are required. System enhancements are intended to provide businesses with dependable and efficient solutions for protecting the integrity and confidentiality of accounting data.

To secure input from unauthorized access and destruction, this paper primarily focuses on data security. It has been said that hybrid cryptography, it combines the Pros of public-key and symmetric-key encoding, an effective way to secure digital data. To ensure that only authorized persons only can access file, it first encrypts it using its symmetric key and encrypts it again using the recipient's public key. The requirement to securely store data has been expanding along with the volume of data, and the rise in malware and phishing attempts has made this issue more important. It also tackles the issue of their local servers' inability to manage high data volumes, encouraging the adoption of cloud computing in spite of its security drawbacks. Researchers have suggested a novel technique to protect important data files after criticizing well-known algorithms like AES, DES, and RSA. (Kr et al., 2019). Sherief H. Murad and Kamel H. Rahouma's book "Hybrid Cryptography for Cloud Security: Methodologies and Designs" [26] aims to investigate how various cryptographic algorithms might be hybridized to improve cloud security. It examines a variety of hybrid cryptological models from 2013 to 2020 and offers information on their applicability, constraints, design, and implementation. It makes the point that symmetric and asymmetric systems work better together and that their security and performance may be enhanced. Although hybrid cryptography has limits in terms of key distribution and computational expense, the research ends with a robust solution for cloud hybrids that provides protection for sensitive data. I would support studies that try to close the gaps and improve Protected measures.

Higher security now requires in all areas are guarantee the confidentiality of the data. A good solution is Robust hybrid encoded, which mix both symmetric and asymmetric cryptography algorithms to encode messages. Symmetric encoded algorithms such as AES and asymmetric algorithms such as RSA, which offer secure key exchange methods, have gained attention due to their effectiveness in encrypting massive datasets. Double encoded technique can be employs the guarantee file security in cloud, according to recent research reports. For example, data security is achieves the use of Protect key management with RSA algorithms and high-speed encryption and decryption techniques like AES. Jaspin et al. (2021) claim that using double encryption with AES and RSA significantly improves data confidentiality and integrity (Matte et al., 2018). By distributing encrypted data among cloud servers, Swarnalatha et al. (2023) improved security by further developing methods for file splitting and encryption2. Key management and overhead computation issues are currently being worked on as open issues. Achieving a favorable performance-security trade-off in optimized hybrid systems will be the main focus of future research, particularly for cutting-edge technologies like blockchain and the Internet of Things.

"Secure File Storage on Cloud using Hybrid Cryptography" (Swathi et al., 2017) is one of them; it examines hybrid cryptography methods to improve cloud storage input text security. 1. The suggested method encrypts and decrypts data using 3DES in conjunction with Blowfish encryption 3. A hybrid approach is used to assure improved security because the technique's recognized weakness is the weakness of a single cryptographic approach. For total protection against unwanted access, data encryption using this suggested paradigm is split into three sections that will be encrypted using various algorithms. 2. Along with reviewing relevant work in the field, this study discusses different encryption methods and their efficiency. There is also a thorough discussion of the specifics of the suggested system's implementation, including all of its advantages in terms of data integrity, very high security, authentication, and confidentiality.

High data security demands that researchers investigate hybrid cryptographic systems. Although it can be difficult, creating hybrid cryptographic systems that combine symmetric and asymmetric encoding methods is necessary for input text security. Asymmetric algorithms like ElGamal and RSA can be used to provide secure key exchange mechanisms, whereas symmetric algorithms like Twofish and AES are well-known for their speed and effectiveness in encrypting big datasets. Research on integrating these capabilities is underway. For example, a hybrid

implementation of RSA and Twofish has been shown to reduce the size of the ciphertext while also producing improved computing performance. AES and RSA work together to provide overall increased security. Indeed, by guaranteeing high encryption levels and effective key management, Kumar et al.'s recent research from 2020 has demonstrated that hybrid cryptosystems offer the potential to significantly improve data security in cloud computing scenarios. Additionally, it has been investigated how to integrate ElGamal with symmetric algorithms to offer security that is resistant to quantum errors. All of these advances have been made, in fact, but the areas that still require research are those that deal with computing overhead and key management complexity. Thus, future advancements would focus on these hybrid systems' scalability and optimal performance(Tajane et al., 2024).

[30] Sadiq Aliyu Ahmad and Ahmed Baita Garko detail how the increasing dependence on cloud storage gives rise to associated problems with regard to security of data. The paper significantly highlights the critical role cryptography plays in securing third-party servers-based data. This reviewer has evaluated various hybrid cryptographic algorithms that emerged recently from 2015 into early 2019. Finally, multiple advantages of combining several encryption techniques for this purpose of improving security will eventually compose their review. It classifies the above literature into comprehensive and tabular reviews, involving common problems like user authentication bypassed and Hybrid Algorithms not implemented properly. The authors conclude that "hybrid cryptography seems to have many security benefits; however, proper user authentication as well as practical implementation of these algorithms could be further improved in cloud computing environments."

This issue of data security in cloud computing systems is examined in the study "A Security Model for the Enhancement of Data Privacy in Cloud Computing," written by Yoshita Sharma, Himanshu Gupta, and Sunil Kumar Khatri [31]. The authors talk about the pressing necessity to protect data privacy and ensure safety. Both consumers and organizations are becoming more and more dependent on cloud services. The study examines the security mechanisms that are currently in place and highlights weaknesses in cloud storage, including data leaks and illegal access. In order to improve data security, it suggests a multi-level encryption paradigm that use of the AES and RSA algorithms. This provides strong protection because data is encrypted at several points, making it impossible for unauthorized parties to avail private information. The authors contend that the aforementioned method can greatly guarantee the security, confidentiality, and integrity of data in cloud computing. The author highlights issues like data lineage and remanence in the paper's discussion of some possible future research topics in cloud security.

3. PROPOSED HYBRID ALGORITHM

The components of symmetric encoding, asymmetric encoding, and hashing are combined in hybrid cryptography, which uses the strength of AES, RSA, and Blake3 to provide safe data protection with minimal performance overhead. The primary data is encrypted at the beginning of the procedure using the symmetric encryption technique of AES. Because AES is a block cipher that maintains very high security levels with little performance loss, it was selected for its high processing speed to encode large volumes of input text.

The AES key itself must next be encoded using an asymmetric algorithm in this case, RSA. RSA employs two keys: a private key for decryption and a public key that can be shared for encryption. Because of this, when the AES key is encoded using RSA, only the recipient who is aware of the private RSA counterpart will be able to decrypt it and obtain the AES key, which will unlock the encrypted data.

The last step is to create a tamper-proof hash of the original data using the cryptography hash function Blake3. Because Blake3 creates a distinct fingerprint of the data, unwanted modification and alteration are easily identified and prevented, allowing integrity testing. Blake3 is thought to be one of the most effective methods for ensuring integrity without sacrificing speed due to its performance in hashing speed and security (Li, 2023).

Together, this hybrid approach strengthens security by transferring, as noted above, the speed of AES, the RSA benefit on key management, and the integrity guarantees of Blake3 into a powerful

and secure cryptographic solution for modern applications.

Table2: Cryptographic Algorithm Characteristics: Key Size and Block Size for AES, RSA, and BLAKE3

Algorithm	Type	KeySize	Block Size
AES	Symmetric Block Cipher	256bits (32 bytes)	128bits(16 bytes)
RSA	Asymmetric (Public-Key)	2048 bits	Variable(depends on key size)
BLAKE3	Cryptographic Hash Function	256 bits	512bits (64bytes)

Once the user invokes the application, it launches, and the user can then select a file to encrypt by using the "Choose File" button. The path selected for the file appears in the entry box. On clicking the "Encrypt File" button, the actual encryption process gets initiated by noting the start time and memory use. It is followed by the AES encoding where salt is randomly generated, and finally, the AES key would be derived with the help of PBKDF2HMAC. The chosen file is encoded using this AES key. The next step is RSA encryption, which involves creating private and public keys and using the RSA public key to encode the AES key. The data and the encoded AES key must then be hashed with BLAKE3. You can choose to save the encrypted file, which will update the encryption time and memory use. While the Tkinter window displays a graph of encryption time versus memory, all of this data is plotted into the GUI.

Similar to decryption, decryption is started by clicking the "Decrypt File" button. A copy of the BLAKE3 hash, which produces the RSA-encrypted AES key and AES-encrypted file contents, is taken after the duration and memory are noted. In order to decode the encoded file contents, the AES key is finally decoded by the RSA private key. The encrypted file is then asked to be saved, and the memory consumption and decryption time are updated and shown. As a result, the Tkinter window displays a graph of the decryption time and memory utilization. As a result, these procedures would finish the encryption process and provide the user with the decryption details.

Flowchart: A multi-step process to encode and decode a file using the combination of the AES, RSA, and BLAKE3 hashing algorithms (Susmitha et al., 2023).

1. Start and Upload File: This is the beginning of the file upload that will be encrypted. It then undergoes several stages of the process of encoding.
2. AES Encoding: The Advanced Encryption Standard algorithm is used to perform the initial level of encryption after the file is uploaded. Often referred to as the first level of encryption, AES encodes file data and produces a partially encrypted output.
3. A Random Key the RSA algorithm is used for RSA encryption. To try to safeguard the AES encryption, the first layer's AES key is encoded using RSA. The second-layer encoding will result from this phase.
4. BLAKE3 Hashing: Next, the AES-encoded file data and the RSA-encoded AES key are hashed using the BLAKE3 hashing function. BLAKE3 is a quick and safe hashing algorithm that strengthens the encryption with an extra layer.
5.Save Encrypted File: Lastly, once the hash is done the last encrypted data is written in a place. Here the encryption is complete and the file is encrypted and safe (Murad & Rahouma, 2022).

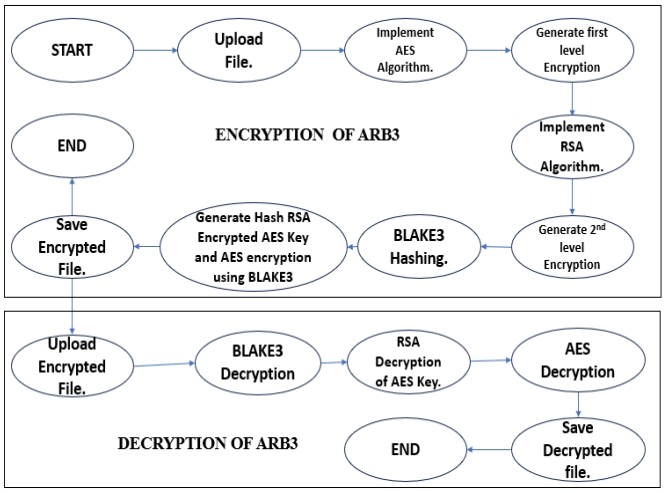

Figure 4: Proposed ARB3 Algorithm Flowchart

6. Decryption Process - Upload encrypted file: First step to decrypt the encrypted file is uploading the encrypted file one more time.
7. BLAKE3 Decryption: Reverse BLAKE3 hashing to recover RSA-encrypted AES key along with AES-encrypted file data
8. RSA Decryption: using private RSA key, it decrypts the AES key with a decryption process that is required for decryption of the file data.
9. AES Decryption: The file data that was encrypted at the first level of encryption is decrypted using

the AES key that has been recovered.
10.Save decrypted file: After that, the decrypted file is saved as an end (Jaspin et al., 2021).
In the whole workflow, multi-layered encryption security could be achieved using symmetric (AES) encryption techniques along with asymmetric (RSA) encryption. These would come in the context of integration with the integrity and hashing capability of BLAKE3. It will enhance data confidentiality and protection against unauthorized access as shown in Figure 4.

Algorithm 1: ARB3 Hybrid Encryption and Decryption

1: INPUT ← Plaintext file F
2: SECRET_KEY ← Static password string
3: SALT ← Static or randomly generated value
4: RSA_Keys ← Generate RSA public/private key pair
5: AES_Key ← Derive using PBKDF2HMAC from SECRET_KEY + SALT
6: Encrypted_F ← AES.encrypt(F, AES_Key)
7: Encrypted_AES_Key ← RSA.encrypt(AES_Key, RSA_Keys.public)
8: Hash ← BLAKE3(Encrypted_F + Encrypted_AES_Key)
9: OUTPUT ← Save (Encrypted_F, Encrypted_AES_Key, Hash) to file

Decryption:
10: INPUT ← (Encrypted_F, Encrypted_AES_Key, Hash) from file
11: AES_Key' ← RSA.decrypt(Encrypted_AES_Key, RSA_Keys.private)
12: Valid ← (BLAKE3(Encrypted_F + Encrypted_AES_Key) == Hash)
13: if Valid:
14: Decrypted_F ← AES.decrypt(Encrypted_F, AES_Key')
15: OUTPUT ← Save Decrypted_F
16: else:
17: OUTPUT ← Integrity verification failed

4. RESULTS OF EXPERIMENTAL RESEARCH

4.1 Experimental Setup

All experiments were conducted under the following hardware and software environment:

- Processor: Intel Core i7-11800H (2.30 GHz, 8 cores)
- RAM: 16 GB DDR4
- OS: Windows 11 64-bit
- Programming Language: Python 3.9
- Libraries: cryptography, psutil, blake3, tkinter
- Dataset: Uniform 4 MB plaintext files (text and log formats)
- Repetitions: Algorithm was tested 5 times, and the average value was taken to ensure consistency and reliability of results.

a. Encryption and Decryption Time in Seconds

Encryption and decryption times are critical metrics that represent the responsiveness and speed of an encryption model. As illustrated in Figure 5, the ARB3 algorithm demonstrates the fastest overall performance, achieving both encryption and decryption in 7 seconds.

Table 3: Encryption and Decryption Time (in seconds)

Algorithm	Encryption Time (s)	Decryption Time (s)
RSA + HMAC	38.02	33.15
RSA + MD5	30.69	30.25
AES + MD5	18.93	14.65
AES + RSA	18.48	13.08
AES + DES + RSA	16.93	11.71
AES + RC6 + Blowfish	17.00	27.00
ARB3 (Proposed)	7.46	7.29

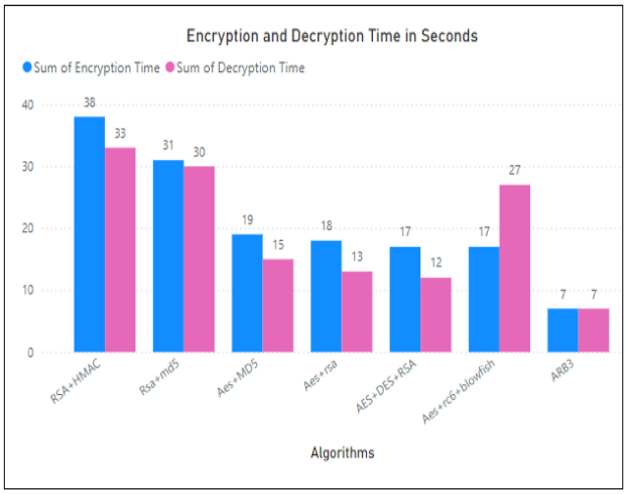

Figure 5: Encryption and decryption time in seconds

b. Memory Consumption during Encryption and Decryption

Figure 6 summarizes the memory consumption (in MB) for encryption and decryption tasks. Each value represents the memory peak recorded during execution using the psutil library.

Table 4: Encryption & Decryption Memory usage

Algorithm	Encryption Memory (MB)	Decryption Memory (MB)
AES + DES + RSA	21.00	16.00
AES + MD5	13.00	16.00
AES + RC6 + Blowfish	14.00	13.00
AES + RSA	12.00	16.00
RSA + HMAC	13.00	16.00
Blowfish + SRNN	13.00	9.00
RSA + MD5	10.00	17.00
ARB3 (Proposed)	16.00	16.00

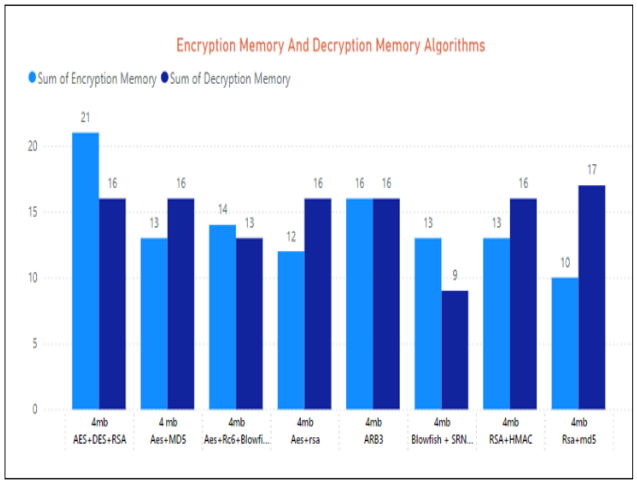

Figure 6: Encryption and decryption memory consumption

c. Throughput

Throughput represents the amount of data that can be processed per second [32]. It is calculated using the formula:

$$Throughput = (\frac{T_P}{E_t + D_t})$$

Where, T_p = Enter plain text (Bytes)

E_t and D_t = Encryption-Decryption Time (seconds)

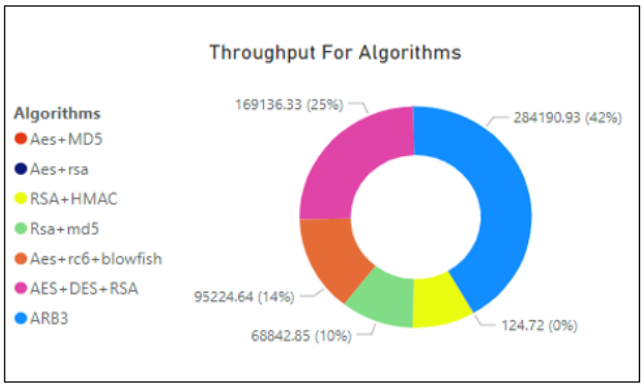

Figure 7: Throughput for Algorithms

Table 5: Throughput for Each Algorithm

Algorithm	Throughput (Bytes/sec)	Share (%)
ARB3 (Proposed)	284,190.93	42%
RSA + HMAC	58,922.00	9%
AES + RC6 + Blowfish	95,224.64	14%
AES + RSA	132,854.74	19%
AES + MD5	124,861.32	18%
RSA + MD5	68,842.85	10%
AES + DES + RSA	16,936.33	2%
Blowfish + SRNN	10,048.57	1%

ARB3 far exceeds everything in its class by a wide margin. The RSA component of ARB3 has little overhead (relative to RSA's cryptographic capabilities of asymmetric encryption). This directly means performance improvements happen by greatly relieving the burden of processing from system overhead of asymmetric encryption (Sharma et al., 2021)
. AES processing is trivial and defines little overhead with high speed. To assist data integrity for consistency and reliability, ARB3 uses the BLAKE3 hashing algorithm which is ultra-fast and designed to scale across multiple cores quickly. ARB3 provides such a powerful and effective hybrid with modern cryptographic features.

5. CONCLUSION AND FUTURE WORK

The proposed hybrid encryption system ARB3 is optimum and convenient in storing and

transmitting data securely. As a hybrid cryptographic algorithm, ARB3 benefits from the expediency of AES to encrypt data, the speed at which RSA is able to run secure key distributions, and the high-speed and parallelity of integrity checks provided by BLAKE3, therefore mitigating the severe weaknesses present in the conventional hybrid encryption schemes, and notably the ones associated with throughput, latency, and system responsiveness. It is evident that experimental outcomes show that ARB3 is better than typical combinations of RSA + HMAC, RSA + MD5, AES + RC6 and Blowfish. It had an encryption time of 7.46 seconds, decryption time of 7.29 seconds and greatest throughput amongst all the tested algorithms of 28,445,023.15 bytes per second and is particularly quality to real-time and high-performance applications.

Centrally, ARB3 is structurally sound and fast enough to find a utility within real-world operations. In cloud storage applications, ARB3 acts to encrypt sensitive files before they can be offloaded to remote servers while BLAKE3 assures the integrity of the data during storage and synchronization. In secure communications, email systems, chat, or file transfer applications, ARB3 can encrypt messages quickly enough to deliver securely and in time. For Virtual Private Networks (VPNs), ARB3 provides a strong cryptographic basis from RSA for negotiating session keys, AES for encrypted tunnels, and BLAKE3 for detecting tampering. And, a low memory profile combined with fast execution speed allows ARB3 to be suitable for IoT devices and edge systems, like industrial sensors and medical devices. In addition, given its high integrity guarantee, ARB3 has a suitable category of applications for government and military systems where secure transfer of data is critical for operational success. Looking to the future, although ARB3 has provided strong results, further work remains. One avenue will involve the integration of post-quantum cryptographic algorithms for hardened security against quantum computed attacks such as Kyber and Dilithium. Another will be the design of lightweight versions of ARB3 for restricted-resource environments such as embedded system and low-power IoT solutions.

Additionally, parallel processing optimizations on multicore processors or GPUs can be explored to further reduce encryption and decryption times. Expanding ARB3's trust model using secure multi-party computation (SMPC) and blockchain-based key verification mechanisms would enhance its application in decentralized systems. In conclusion, ARB3 successfully bridges the gap in current hybrid encryption approaches by offering a high-performance, secure, and scalable solution. It demonstrates strong potential as a next-generation cryptographic system suitable for diverse domains including cloud computing, network security, embedded systems, and governmental infrastructures. Its implementation stands as a future-ready strategy for safeguarding sensitive information in today's interconnected digital world.

REFERENCES

Sharma, H., Kumar, R., & Gupta, M. (2023). A review paper on hybrid cryptographic algorithms in cloud network. 2023 2nd International Conference for Innovation in Technology (INOCON), 1–5. https://doi.org/10.1109/INOCON57975.2023.10101044

Fernando, E., Agustin, D., Irsan, M., Murad, D. F., Rohayani, H., & Sujana, D. (2019). Performance comparison of symmetries encryption algorithm AES and DES with Raspberry Pi. 2019 International Conference on Sustainable Information Engineering and Technology (SIET), 353–357. https://doi.org/10.1109/SIET48054.2019.8986122

Zhang, Q. (2021). An overview and analysis of hybrid encryption: The combination of symmetric encryption and asymmetric encryption. 2021 2nd International Conference on Computing and Data Science (CDS), 616–622. https://doi.org/10.1109/CDS52072.2021.00111

Meng, Z., & Wang, Y. (2022). Asymmetric encryption algorithms: Primitives and applications. 2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI), 876–881. https://doi.org/10.1109/ICETCI55101.2022.9832032

D. P., Babu, S. S., & Vijayalakshmi, Y. (2020). Enhancement of e-commerce security through asymmetric key algorithm. Computer Communications, 153, 125–134. https://doi.org/10.1016/j.comcom.2020.01.033

Jirwan, N., Singh, A., & Vijay, S. (2019). Review and analysis of cryptography techniques. International Journal of Scientific Engineering Research, 4(3), 1–6.

Timothy, D. P., & Santra, A. K. (2017). A hybrid cryptography algorithm for cloud computing security. 2017 International Conference on Microelectronic Devices, Circuits and Systems (ICMDCS), 1–5. https://doi.org/10.1109/ICMDCS.2017.8211728

Kumar, A., Jain, V., & Yadav, A. (2020). A new approach for security in cloud data storage for IoT applications using hybrid cryptography technique. 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC), 514–517. https://doi.org/10.1109/PARC49193.2020.236666

Hyseni, D., Luma, A., & Selimi, B. (2018). The proposed model to increase security of sensitive data in cloud computing. International Journal of Advance Research in Engineering, Science and Technology, 9(2).

Lu, C. C., & Tseng, S. Y. (2002). Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter. Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors, 277–285.

Maitri, P. V., & Verma, A. (2016). Secure file storage in cloud computing using hybrid cryptography algorithm. 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 1635–1638. https://doi.org/10.1109/WiSPNET.2016.7566416

Kumar, S., Karnani, G., Gaur, M., & Mishra, A. (2021). Cloud security using hybrid cryptography algorithms. In 2021 International Conference on Innovative Engineering and Management (ICIEM) (pp. 599–604). https://doi.org/10.1109/ICIEM51511.2021.9445377

Aufa, F. J., Endroyono, & Affandi, A. (2018). Security system analysis in combination method: RSA encryption and digital signature algorithm. 2018 4th International Conference on Science and Technology (ICST), 1–5. https://doi.org/10.1109/ICSTC.2018.8528584

Ciocan, I. T., Kelesidis, E. A., Maimuţ, D., & Morogan, L. (2021). A modified Argon2i using a tweaked variant of Blake3. 2021 26th IEEE Asia-Pacific Conference on Communications (APCC), 271–274. https://doi.org/10.1109/APCC49754.2021.9609933

Kahri, F., Bouallegue, B., Machhout, M., & Tourki, R. (2013). An FPGA implementation of the SHA-3: The BLAKE hash function. 2013 10th International Multi-Conference on Systems, Signals & Devices (SSD13), 1–5. https://doi.org/10.1109/SSD.2013.6564030

Baqtian, H., & Al-Aidroos, N. (2023). Three hash functions comparison on digital Holy Quran integrity verification. Journal of Computer and Information Technology, 11, 1–7.

Verma, V., Kumar, P., Verma, R. K., & Priya, S. (2021). A novel approach for security in cloud data storage using AES-DES-RSA hybrid cryptography. 2021 Emerging Trends in Industry 4.0 (ETI 4.0), 1–6. https://doi.org/10.1109/ETI4.051663.2021.9619274

Chinnasamy, P., & Deepalakshmi, P. (2018). Design of secure storage for health-care cloud using hybrid cryptography. 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 1717–1720. https://doi.org/10.1109/ICICCT.2018.8473107

Gajendra, B. P., Singh, V. K., & Sujeet, M. (2016). Achieving cloud security using third party auditor, MD5 and identity-based encryption. 2016 International Conference on Computing, Communication and Automation (ICCCA), 1304–1309. https://doi.org/10.1109/CCAA.2016.7813920

Kr, S., Babu, S., & Vijayalakshmi, Y. (2019). Enhancing the security of cloud data using hybrid encryption algorithm. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-019-01403-1

Matte, S., Dubey, A., Shirsat, N., & Kale, A. (2018). Hybrid model for securing e-commerce transaction. International Journal of Scientific Engineering Research, 9(4), 25–26.

Swathi, B., Bhaludra, S. D., & Raveendranadh, S. (2017). Secure file storage in cloud computing using hybrid cryptography algorithm. International Journal of Advance Research in Science and Engineering, 6(11).

Tajane, K., Pitale, R., Zambre, S., Huda, H., Utage, A., & Dhar, V. (2024). Efficient cloud data deduplication with Blake3 and secure transfer using AES. 2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN), 572–579. https://doi.org/10.1109/ICPCSN62568.2024.00096

Li, Z. (2023). Exploration on accounting data encryption processing system based on DES algorithm. 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), 1–6. https://doi.org/10.1109/AIKIIE60097.2023.10389923

Susmitha, C., Srineeharika, S., Laasya, K. S., Kannaiah, S. K., & Bulla, S. (2023). Hybrid cryptography for secure file storage. 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), 1151–1156. https://doi.org/10.1109/ICCMC56507.2023.10084073

Murad, S., & Rahouma, K. (2022). Hybrid cryptography for cloud security: Methodologies and designs. In Emerging Trends in Cybersecurity (pp. 95–112). Springer. https://doi.org/10.1007/978-981-16-2275-5_7

Jaspin, K., Selvan, S., Sahana, S., & Thanmai, G. (2021). Efficient and secure file transfer in cloud through double encryption using AES and RSA algorithm. 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), 791–796. https://doi.org/10.1109/ESCI50559.2021.9397005

Sharma, V., Chauhan, A., Saxena, H., Mishra, S., & Bansal, S. (2021). Secure file storage on cloud using hybrid cryptography. 2021 5th International Conference on Information Systems and Computer Networks (ISCON), 1–6. https://doi.org/10.1109/ISCON52037.2021.9702323

Jintcharadze, E., & Iavich, M. (2020). Hybrid implementation of Twofish, AES, ElGamal and RSA cryptosystems. 2020 IEEE East-West Design & Test Symposium (EWDTS), 1–5. https://doi.org/10.1109/EWDTS50664.2020.9224901

Ahmad, S. A., & Garko, A. B. (2019). Hybrid cryptography algorithms in cloud computing: A review. 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), 1–6. https://doi.org/10.1109/ICECCO48375.2019.9043254

Sharma, Y., Gupta, H., & Khatri, S. K. (2019). A security model for the enhancement of data privacy in cloud computing. 2019 Amity International Conference on Artificial Intelligence (AICAI), 898–902. https://doi.org/10.1109/AICAI.2019.8701398

International Journal of Mathematical Sciences and Computing. (2020). An approach to secure cloud computing with hybrid encryption (Vol. 4, pp. 35–41). MECS. https://doi.org/10.5815/ijmsc.2020.04.04