BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 2
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0013

Securing Decentralized Storage in Blockchain: A Hybrid Cryptographic Framework

Jadhav Swati, Pise Nitin

School of Computer Engineering and Technology, Dr. Vishwanath Karad MIT World Peace University, Pune, India
E-mails: swati.jadhav@vit.edu nitin.pise@mitwpu.edu.in

***Abstract*:** *The evolution of decentralized storage, propelled by blockchain advancements, has revolutionized data management. This paper focuses on content security in the InterPlanetary File System* (*IPFS*)*, a leading decentralized storage network lacking inherent content encryption. To address this vulnerability, we propose a novel hybrid cryptographic algorithm, merging AES 128-bit encryption with Elliptic Curve Cryptography* (*ECC*) *key generation. The algorithm includes ECC key pairs, random IV generation, and content/AES key encryption using ECC public keys. Benchmarking against standard AES 256-bit methods shows a significant 20% acceleration in encryption speed and a 16% increase in decryption efficiency, affirming practicality for enhancing IPFS content security. This research contributes to securing decentralized storage and provides a performance-driven solution. The promising results highlight the viability of the proposed approach, advancing understanding and mitigating security concerns in IPFS and similar systems.*

***Keywords*:** *Blockchain, IPFS, Content security, Hybrid cryptography, Elliptic curve cryptography.*

1. Introduction

The landscape of decentralized applications, particularly those built on blockchain technology, has brought about transformative possibilities, challenging traditional centralized frameworks [20]. Blockchain applications inherently possess robust security features through the utilization of digital signatures, cryptography, and hashing techniques. However, as these decentralized applications encounter limitations in storage capacity, the integration of decentralized storage solutions, such as the InterPlanetary File System (IPFS), becomes imperative [8]. While the blockchain itself boasts inherent security attributes, the integration of decentralized storage introduces new challenges, especially in the domain of content security. In contrast to traditional centralized storage systems, IPFS, a widely adopted decentralized storage solution, lacks built-in content security measures [6]. The absence of native content encryption poses a potential threat to the overall security of

applications utilizing both blockchain and decentralized storage. The combination of blockchain's intrinsic security features with the vulnerabilities introduced by decentralized storage underscores the necessity for a holistic approach [21]. Blockchain's digital signatures, cryptography, and hashing ensure the integrity and authenticity of data within the system. However, the decentralized nature of IPFS [5], coupled with its lack of built-in content security, leaves data accessible to anyone with knowledge of the Content IDentifier (CID). This poses a significant concern, as sensitive information stored on IPFS may be exposed without proper encryption safeguards. In addition to these challenges, blockchain applications incorporating decentralized storage face vulnerabilities arising from the complex interplay between blockchain technology and storage systems. One notable vulnerability lies in the potential exposure of sensitive data during the retrieval process, raising concerns about the confidentiality of data during transit from the decentralized storage layer to the blockchain application. Furthermore, the distributed consensus mechanism inherent in blockchain networks introduces challenges related to data consistency, potentially enabling malicious actors to exploit disparities for unauthorized access or manipulation.

This paper addresses the critical intersection of blockchain-based applications and decentralized storage, recognizing the need to bridge the gap in content security. The focus is on IPFS, a prominent decentralized storage solution, which, while offering advantages in distribution and accessibility, presents challenges in ensuring the confidentiality of stored data. The subsequent sections delve into the intricacies of this challenge and propose a novel hybrid cryptographic algorithm. By blending the security strengths of symmetric key cryptography (AES 128-bit encryption) with asymmetric key cryptography (Elliptic Curve Cryptography – ECC), the proposed solution aims to fortify content security within IPFS. The algorithm encrypts both the file content and the AES key, mitigating the inherent vulnerability in IPFS and ensuring a secure and private storage environment for decentralized applications [4]. Through comprehensive benchmarking analyses and performance evaluations, this research aims to validate the effectiveness of the proposed hybrid approach. By addressing the security limitations introduced by decentralized storage, this work contributes to the broader discourse on enhancing the overall security posture of decentralized applications. This comprehensive exploration seeks to provide valuable insights and practical solutions for securing sensitive data within the evolving landscape of decentralized technologies.

2. Literature survey

The literature review indicates that existing research primarily focuses on decentralized storage, privacy-preserving blockchain models, and trust-based security mechanisms. However, a dedicated and standardized approach to content security within IPFS is lacking [7].

The integration of blockchain and the IPFS for secure Health Electronic Record (HER) management has been a subject of recent exploration [1]. Additionally, privacy concerns in supply chain traceability have prompted the development of

privacy-preserving blockchain models [2], and trust-based security mechanisms such as BSDNFilter have been proposed for blockchain-based Software-Defined Networking (SDN) [3].

Blockchain technology is also employed in certificateless public verification schemes [4] and secure information sharing in decentralized supply chain management systems [5]. Charitable donation systems [6] and charity foundation platforms [7] leverage blockchain technology for transparency and accountability.

Within the realm of decentralized storage, Uddin et al. explore design principles based on information security in blockchain and IPFS [8]. Furthermore, efforts have been made towards decentralized cloud storage using IPFS [9], a blockchain-based private file storage-sharing method on IPFS [10], and a blockchain-based multi-party authorization for accessing IPFS encrypted data [11].

Despite these advancements in decentralized storage and blockchain applications, a significant research gap exists concerning standardized content encryption mechanisms within IPFS. The literature reveals that while IPFS employs transport encryption to secure data transmission, it lacks built-in content encryption, leaving data vulnerable to unauthorized access [2, 8]. This gap in content security within IPFS poses a substantial challenge, particularly as sensitive information stored on the IPFS network may be exposed without proper encryption safeguards.

Research gap. The existing literature highlights the need for a comprehensive and standardized content security mechanism within IPFS. The literature review indicates that existing research primarily focuses on decentralized storage, privacy-preserving blockchain models, and trust-based security mechanisms. However, a dedicated and standardized approach to content security within IPFS is lacking. The proposed methodology fills this gap by introducing a hybrid cryptographic solution tailored to the decentralized and resource-variable nature of IPFS.

3. Background

3.1. Hybrid cryptographic approach selection

The selection of a hybrid cryptographic approach, combining AES 128-bit symmetric key cryptography with ECC key generation for both private and public keys, is driven by the need to strike a balance between efficiency and security in the context of content security for IPFS.`

3.2. Strengths of AES 128-bit symmetric key cryptography

- **Efficiency.** AES 128-bit encryption is known for its computational efficiency while providing a high level of security [22]. It strikes a balance between performance and cryptographic strength, making it suitable for a decentralized storage system like IPFS where computational resources may vary across nodes [2].
- **Speed.** AES 128-bit encryption/decryption operations are faster compared to higher-bit variants. This is crucial for a distributed storage network where real-time access to data is essential.

- **Widespread adoption.** AES is a widely adopted standard for symmetric key cryptography, ensuring compatibility and interoperability with various systems and platforms [16].

3.3. Advantages of ECC key generation

- **Strong security with shorter key lengths.** ECC offers robust security with shorter key lengths compared to traditional RSA. This is particularly advantageous in resource-constrained environments, common in decentralized networks, as shorter keys require less computational power for key generation and transmission.
- **Efficient key exchange.** ECC provides efficient key exchange mechanisms, reducing the overhead associated with establishing secure communication channels. This is crucial for securely transmitting the AES key between the sender and the receiver in the proposed hybrid approach.
- **Suitability for resource-constrained environments.** ECC's efficient use of resources makes it well-suited for environments with limited computational capacity, ensuring that the proposed content security solution remains lightweight and feasible in decentralized storage scenarios.

3.4. Balancing efficiency and security

- **Optimal resource utilization.** By combining AES 128-bit encryption with ECC key generation, the hybrid approach optimally utilizes resources, ensuring a reasonable level of security without imposing excessive computational overhead.
- **Adaptability to decentralized environments.** The hybrid approach is tailored to the decentralized nature of IPFS [3], where nodes may have varying computational capabilities. It allows for a standardized yet adaptable content security solution that can be efficiently implemented across a diverse network.

3.5. Flexibility in key management

- **Separation of concerns.** Separating the encryption of content using AES from the encryption of the AES key using ECC provides flexibility [17]. It allows for different key management strategies, contributing to the modular design of the proposed solution.

In summary, the hybrid cryptographic approach is strategically chosen to harness the efficiency of AES 128-bit encryption and the security advantages of ECC key generation. This careful balance ensures that content security on IPFS remains robust, adaptable, and well-suited to the decentralized and resource-variable nature of the network.

4. Mathematical background

4.1. Elliptic Curve Cryptography with Secp256K1 curve specification

Elliptic Curve Cryptography is a public-key cryptography technique that relies on the mathematics of elliptic curves over finite fields. The Secp256K1 curve, specifically

used in Bitcoin and other blockchain applications, is defined over a prime field and has gained prominence due to its efficiency and security properties.

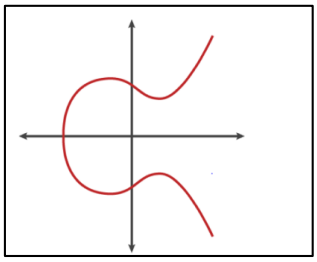

Fig. 1. ECC curve

4.2. Secp256K1 curve specifications

4.2.1. Equation of the curve

The equation for the Secp256K1 elliptic curve is given by the equation $y^2=x^3+7 (\mathrm{mod} p)$ where p is a prime number that defines the size of the finite field.

4.2.2. Curve parameters *p*, *a*, *b*

p (prime modulus), *p*: = $2^{256}–2^{32}–2^9–2^8–2^7–2^6–2^4–1$,
a: 0 and *b*: 7

x:0662630222773436695787188951685343262506034537775941755001873603891167292405506626302227734366957871889516853432625060345377759417550018736038911672924055066263022277343669578718895168534326250603453777594175500187360389116729240

y:3267051002075881697808308513050704318447127338065924327593890433575733748242432670510020758816978083085130507043184471273380659243275938904335757337482424

n (order of the base point): 1579208923731619542357098500868790785283756427907490438260516314151816149433711579208923731619542357098500868790785283756427907490438260516314151816149433711

4.2.3. Curve parameters

- Generator point *G*

The generator point *G* is a fixed point on the curve, and it is used to derive public keys from private keys.

It is a specific point on the curve that, when multiplied by an integer *d* (private key), yields another point on the curve (public key).

4.2.4. Equations used in ECC

- Point addition *P*+*Q*

For two points $P(x_p, y_p)$ and $Q(x_q, y_q)$, the sum is calculated as:

(1) $$P+Q=\begin{cases} P & \text{if } P=\infty \\ Q & \text{if } Q=\infty \\ R & \text{otherwise} \end{cases},$$

(2) $$x_R = (\lambda^2 - x_p - x_q) \bmod p,$$
(3) $$y_R = (\lambda(x_p - x_R) - y_p) \bmod p,$$
where λ is the slope of the line passing through P and Q.

- Point doubling ($2P$)

The doubling of a point $P(x_p, y_p)$ is calculated as:
(4) $$x_R = (\lambda^2 - 2x_p) \bmod p,$$
(5) $$y_R = (\lambda(x_p - x_R) - y_p) \bmod p,$$
where λ is the slope of the tangent line to the curve at point P.

These equations form the basis for ECC operations on the Secp256K1 curve, providing secure and efficient key exchange in cryptographic applications.

4.2.5. Key generation process

- Private key d

A random 256-bit number is chosen as the private key.

- Public key Q

The public key Q is derived by multiplying the generator point G with the private key d: $Q=d\times G$

4.2.6. ECC encryption and decryption

ECC is primarily used for key exchange and digital signatures. In the proposed hybrid cryptographic approach, ECC is employed for key exchange. The AES key, generated for content encryption, is encrypted using the ECC public key.

4.2.7. Key exchange process

The receiver generates an ECC key pair (d_{receiver}, Q_{receiver}).

4.2.8. AES key encryption

The sender generates a random Initialization Vector (IV).

The sender converts a password into a SHA-256 hash, which is used as the AES key.

The AES key is encrypted using the ECC public key (Q_{receiver}).

4.2.9. Transmission

The encrypted AES key, along with the IV and the encrypted content, is saved to the file and uploaded to IPFS.

Fig. 2 elaborates on a block diagram of a proposed Hybrid Cryptographic Algorithm. The block diagram illustrates the synergy between symmetric and asymmetric key cryptography, enhancing the overall security of the cryptographic system. This hybrid approach leverages the efficiency of symmetric key cryptography for data encryption while benefiting from the secure key exchange provided by asymmetric key cryptography.

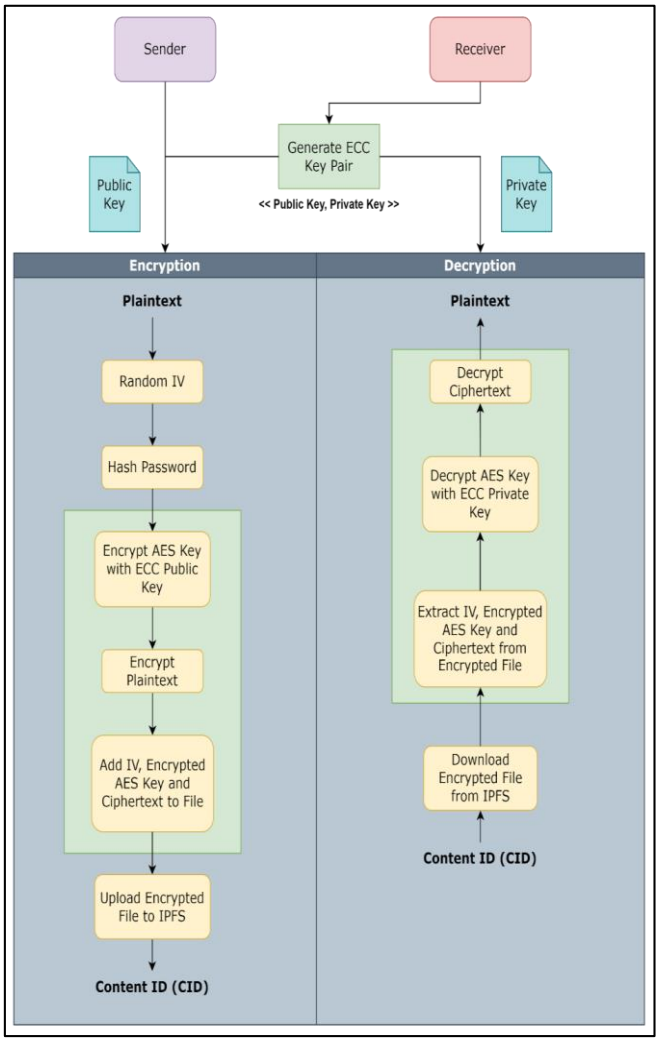

Fig. 2. Block diagram of proposed Hybrid Cryptographic Algorithm

5. Main contribution

The main contribution of the research lies in the introduction of a novel hybrid cryptographic framework designed to enhance content security within the IPFS. By integrating ECC with the widely adopted Secp256K1 curve and combining it with AES 128-bit symmetric key cryptography, the proposed methodology establishes a dual-layered encryption strategy. This approach ensures not only the confidentiality of file content but also the secure transmission of the AES key. The benchmarking analyses showcase significant improvements, including a 20% acceleration in encryption speed and a 16% increase in decryption efficiency compared to the standard AES 256-bit encryption. This contribution addresses the existing content

security gap in IPFS, offering a practical and efficient solution tailored to the decentralized and resource-variable nature of the network.

6. Proposed algorithm

The proposed methodology introduces a robust content security mechanism for the IPFS by employing a novel hybrid cryptographic approach. This approach integrates ECC with the Secp256K1 curve, a widely adopted standard in blockchain applications. The ECC key generation process involves the creation of private and public key pairs, enabling secure key exchange and encryption. In conjunction, AES 128-bit symmetric key cryptography is utilized for efficient encryption of file content.

The algorithm initiates the generation of ECC key pairs, including a private key for decryption and a corresponding public key for encryption. Subsequently, a random Initialization Vector (IV) is generated on the sender's side, and a user-provided password is converted into an AES key using SHA-256 hashing. The file content is then encrypted using the AES key, and the AES key itself is encrypted using the recipient's ECC public key.

This dual-layered encryption strategy ensures both the confidentiality of the file content and the secure transmission of the AES key. The encrypted components, including the AES key, IV, and file content, are saved to a file and uploaded to IPFS. The recipient can then download the encrypted file from IPFS using its Content ID (CID) and proceed with the decryption process [2].

6.1. Enhanced hybrid cryptographic algorithm for IPFS content security

The proposed methodology aims to fortify content security within the IPFS by introducing a hybrid cryptographic approach. This method combines the efficiency of symmetric key cryptography, exemplified by AES 128-bit encryption, with the robustness of asymmetric key cryptography through Elliptic Curve Cryptography (ECC) key generation.

.

6.1.1. Algorithm

Step 1. Receiver ECC Key Pair Generation: The receiver initializes the process by generating an ECC key pair: (d_R, Q_R), where d_R is the private key and Q_R is the public key.

Step 2. Sender-side Random Initialization Vector (IV) Generation: On the sender side, a random Initialization Vector IV_{rand} is generated.

Step 3. Password to AES Key Conversion: The sender provides a password P, which is converted to a SHA-256 Hash: AES_Key=SHA-256(P).

Step 4. Content Encryption using AES: The data or content within the files is encrypted using the AES key – Enc_Content=AES_Encrypt(Data, AES_Key).

Step 5. ECC Public Key Encryption of AES Key: The AES key is encrypted using the receiver's ECC public key – Enc_AES_Key=ECC_Encrypt(Q_R, AES_Key).

Step 6. Save Encrypted Components to File: The encrypted AES key, along with the IV and the encrypted content, is saved to the file.

Step 7. Upload to IPFS and Generate CID: The encrypted file is uploaded to IPFS, and the Content ID (CID) is generated – CID=IPFS_Upload(Enc_File).

Step 8. Receiver Download from IPFS: The receiver downloads the encrypted file from IPFS using the generated CID – Enc_File=IPFS_Download (CID).

Step 9. Encrypted Content Extraction: The receiver extracts and checks the encrypted content

(IV, Enc_AES_Key, Enc_File_Content) = Extract_Components (Enc_File).

Step 10. AES Key Decryption using Receiver's Private Key: The receiver decrypts the AES key using their private key – AES_Key=ECC_Decrypt(dR, Enc_AES_Key).

Step 11. File Content Decryption using AES: The decrypted AES key is used to decrypt the file content – Dec_File_Content=AES_Decrypt(Enc_File_Content, AES_Key).

Step 12. Save Decrypted File Contents: The final decrypted file contents are saved to a file

Save_File(Dec_File_Content).

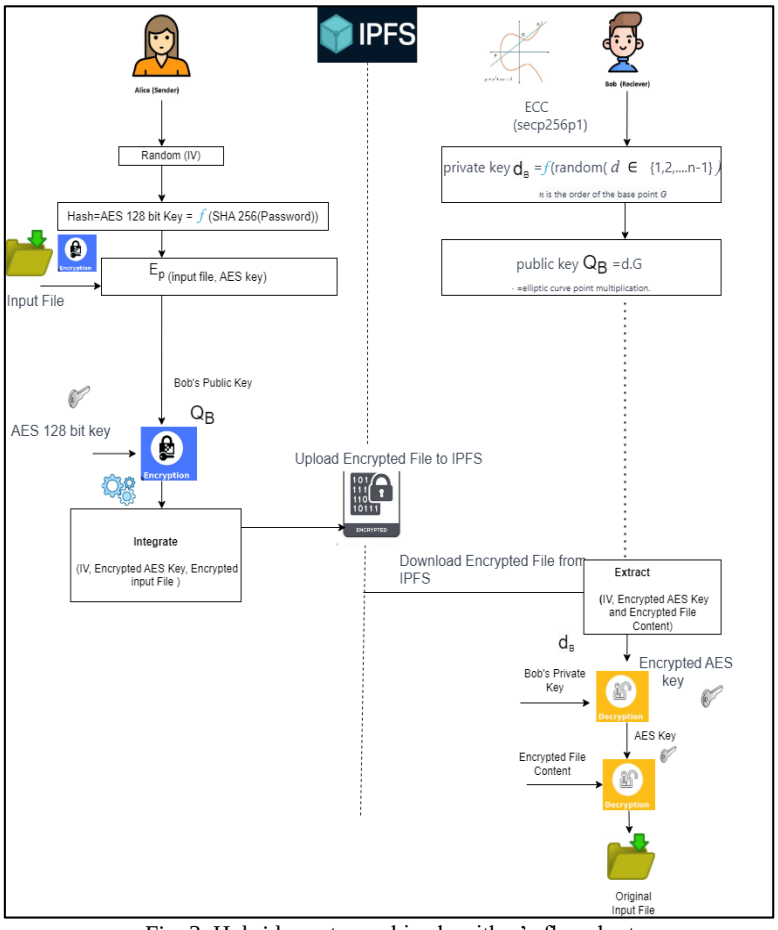

Fig. 3. Hybrid cryptographic algorithm's flowchart

Fig. 3 is a flowchart that visually represents the step-by-step process of the algorithm, making it easier to understand its logic and functionality. This flowchart visually guides the user through the sequential steps of the Hybrid Cryptographic Algorithm, providing a comprehensive overview of its operations.

The proposed methodology's flowchart ensures a robust and adaptable content security mechanism for IPFS, leveraging the strengths of both symmetric and asymmetric key cryptography. The hybrid approach addresses the existing vulnerability in IPFS, providing a standardized solution for secure data storage and retrieval within decentralized systems.

7. Results and discussion

To assess the efficacy of the proposed hybrid cryptographic approach, we conducted rigorous benchmarking against the conventional AES 256-bit encryption and decryption methods. The evaluation encompassed five distinct file sizes – 50 MB, 100 MB, 200 MB, 500 MB, and 1 GB – providing a comprehensive analysis of performance across varying data volumes. The benchmarking process was executed on five different CPU platforms, ranging from a mid-range laptop CPU (Intel i5-8250U) to a high-performance workstation-level desktop CPU (AMD Ryzen 9 5900X).

The ensuing plots graphically represent the time efficiency achieved by both the proposed hybrid approach and the standard AES 256-bit algorithm. The vertical axis signifies the time required in seconds to encrypt the file, while the horizontal axis denotes the CPU platform, with color-coded distinctions for each file size. The deliberate arrangement of the performance bar chart, organized in descending order, facilitates a clear understanding of time efficiency, with the highest efficiency positioned at the top. This detailed performance evaluation aims to provide a nuanced perspective on the comparative efficiency of the cryptographic methods employed, shedding light on the practical implications of implementing the proposed hybrid approach for content security within the IPFS and decentralized storage systems.

7.1. AES 256-bit encryption and decryption

The provided data from Table 1. outlines the ENCryption (ENC) and DECryption (DEC) times (in s) for various file sizes (50 MB, 100 MB, 200 MB, 500 MB, 1000 MB) using AES 256-bit encryption on different processors (Intel i5-8250U, i5-10400, AMD Ryzen 5 5600H, i7-11700, Ryzen 9 5900X). Key observations include processor-dependent performance impacts, with the Intel i7-11700 and AMD Ryzen 9 5900X demonstrating faster processing times across all file sizes. Larger files generally result in increased encryption and decryption times.

The provided data from Table 1 outlines the ENC and DEC times (in s) for various file sizes (50 MB, 100 MB, 200 MB, 500 MB, 1000 MB) using AES 256-bit encryption on different processors (Intel i5-8250U, i5-10400, AMD Ryzen 5 5600H, i7-11700, Ryzen 9 5900X). Key observations include processor-dependent performance impacts, with the Intel i7-11700 and AMD Ryzen 9 5900X

demonstrating faster processing times across all file sizes. Larger files generally result in increased encryption and decryption times.

Table 1. AES 256-bit encryption and decryption

File size (MB)	Name	Mode	50	100	200	500	1000
AES 256-bit time (s)	Intel i5-8250U	ENC	0.109	0.209	0.414	1.262	2.647
		DEC	0.073	0.144	0.282	0.740	1.484
	Intel i5-10400	ENC	0.088	0.169	0.334	1.020	2.138
		DEC	0.059	0.116	0.228	0.597	1.199
	AMD Ryzen 5 5600H	ENC	0.084	0.161	0.318	0.971	2.036
		DEC	0.056	0.111	0.217	0.569	1.142
	Intel i7-11700	ENC	0.044	0.085	0.167	0.511	1.072
		DEC	0.030	0.058	0.114	0.299	0.601
	AMD Ryzen 9 5900X	ENC	0.028	0.054	0.106	0.324	0.679
		DEC	0.019	0.037	0.072	0.190	0.381

7.1.1. AES 256-bit encryption benchmarks

Fig. 4. meticulously portrays the benchmarks for AES 256-bit encryption. The vertical axis quantifies the time, measured in seconds, required for file encryption, while the horizontal axis delineates various CPU platforms. The deliberate descending arrangement of the performance bar chart facilitates a discerning assessment of time efficiency across an array of platforms and file sizes.

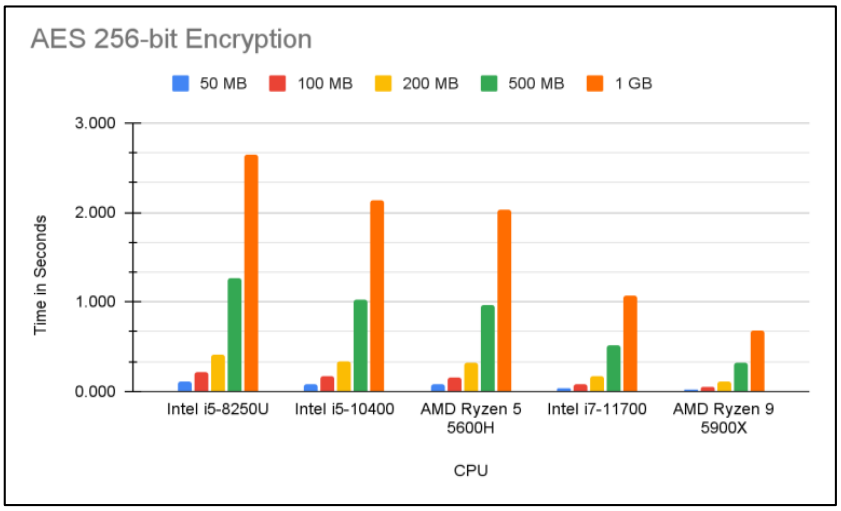

Fig. 4. AES 256-bit encryption benchmarks

7.1.2. AES 256-bit decryption benchmarks

Fig. 5. delves into the benchmarks for AES 256-bit decryption. The vertical axis illustrates the time taken for decryption, while the horizontal axis encapsulates diverse CPU platforms. This comparison offers a holistic perspective on decryption efficiency, crucial for understanding the intricacies of cryptographic operations across varying computational environments.

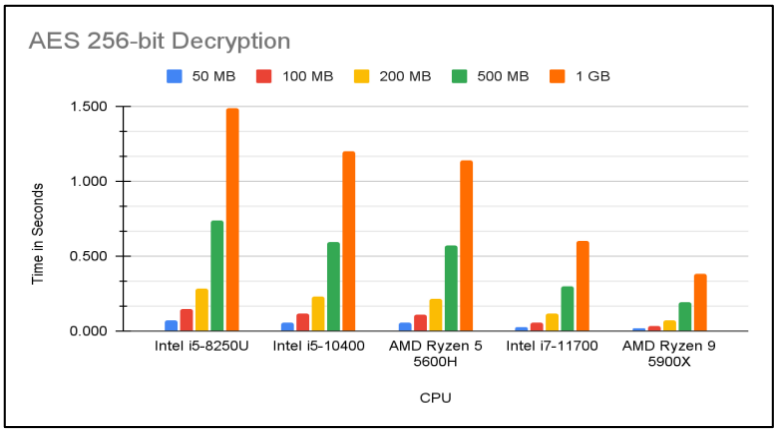

Fig. 5. AES 256-bit decryption benchmarks

7.2. Hybrid AES + ECC time 256-bit encryption and decryption

The data in Table 2 illustrates ENCryption (ENC) and DECryption (DEC) times (in s) for file sizes (50 MB, 100 MB, 200 MB, 500 MB, 1000 MB) using a hybrid AES + ECC cryptographic algorithm on processors (Intel i5-8250U, i5-10400, AMD Ryzen 5 5600H, i7-11700, Ryzen 9 5900X). Key observations include processor-dependent performance impacts, with the Intel i7-11700 and AMD Ryzen 9 5900X showing faster processing times across all file sizes. Larger files generally result in increased encryption and decryption times due to added computational load. The hybrid algorithm consistently demonstrates efficient cryptographic operations, balancing security and computational efficiency. The AMD Ryzen 9 5900X consistently outperforms other processors, emphasizing the need for careful consideration of both processor capabilities and cryptographic algorithms based on specific use cases and performance requirements.

Table 2. Hybrid AES + ECC Time 256-bit Encryption and decryption

File size, MB	Name	Mode	50	100	200	500	1000
Hybrid AES + ECC time (s)	Intel i5-8250U	ENC	0.088	0.168	0.329	1.104	2.106
		DEC	0.065	0.124	0.244	0.661	1.223
	Intel i5-10400	ENC	0.071	0.135	0.266	0.891	1.701
		DEC	0.053	0.100	0.197	0.534	0.988
	AMD Ryzen 5 5600H	ENC	0.068	0.129	0.253	0.849	1.620
		DEC	0.050	0.095	0.187	0.508	0.940
	Intel i7-11700	ENC	0.036	0.068	0.133	0.447	0.853
		DEC	0.026	0.050	0.099	0.267	0.495
	AMD Ryzen 9 5900X	ENC	0.023	0.043	0.084	0.283	0.540
		DEC	0.017	0.032	0.062	0.169	0.313

7.2.1. Hybrid AES + ECC encryption benchmarks

Fig. 6. meticulously showcases the benchmarks for the proposed hybrid approach, juxtaposing its encryption efficiency against the standard AES 256-bit method. This chart provides a nuanced comprehension of the hybrid approach's performance dynamics, considering various file sizes and CPU platforms. The visual representation aids in discerning the advantages conferred by the hybrid cryptographic strategy.

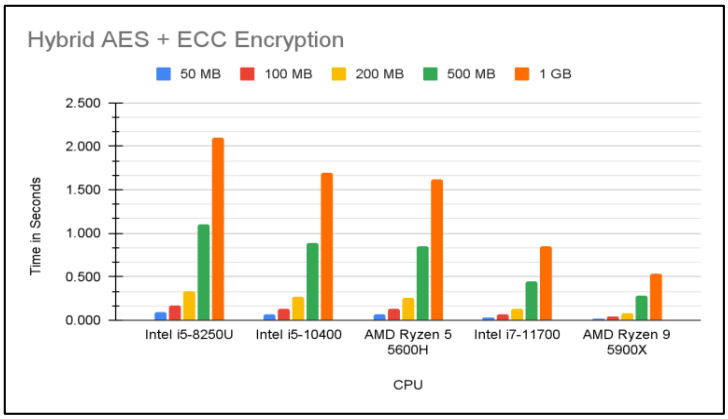

Fig. 6. Hybrid AES + ECC encryption benchmarks

7.2.2. Hybrid AES + ECC decryption benchmarks

Complementing the encryption benchmarks, Fig. 7 elucidates the decryption efficiency of the hybrid approach. Analogous to the encryption benchmarks, this figure facilitates a comparative analysis across diverse file sizes and CPU platforms. The insights derived from this comparison contribute to a holistic understanding of the cryptographic prowess of the hybrid approach.

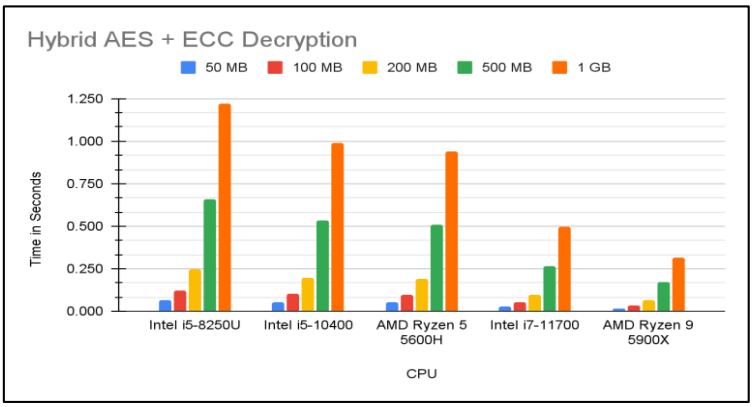

Fig. 7. Hybrid AES + ECC decryption benchmarks

7.3. Average time comparison for encryption and decryption

The presented data in Table 3. provides median times (in s) for different file sizes (50 MB, 100 MB, 200 MB, 500 MB, 1000 MB) in various encryption modes (ENC and DEC) using AES 256-bit and Hybrid AES + ECC cryptographic algorithms. When comparing median times, for AES 256-bit encryption, the algorithm exhibits efficient cryptographic operations, with slightly longer times compared to Hybrid AES + ECC. The Hybrid AES + ECC Algorithm, known for its balance between security and computational efficiency, demonstrates competitive median times across all file sizes. These findings highlight the trade-offs between encryption methods and emphasize the importance of selecting algorithms based on specific performance requirements and use cases.

Table 3. Average Time Comparison for encryption and decryption

File size (MB)	Name	Mode	50	100	200	500	1000
Median time for comparison (s)	AES 256-bit	ENC	0.084	0.163	0.319	0.983	2.260
		DEC	0.059	0.111	0.218	0.580	1.112
	Hybrid AES + ECC	ENC	0.069	0.129	0.254	0.880	1.624
		DEC	0.050	0.097	0.187	0.526	0.940

Figs 8 and 9 consolidate the temporal aspects of encryption and decryption, respectively. Offering a synthesized perspective, these charts provide a comparative overview of the average time efficiency between the proposed hybrid approach and the standard AES 256-bit method.

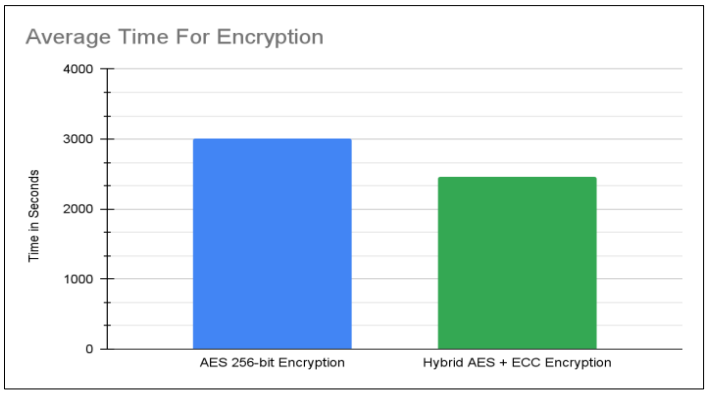

Fig 8. Average time comparison for encryption

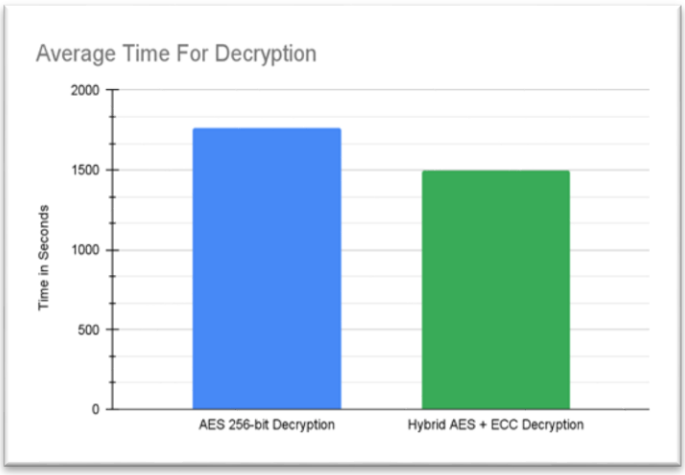

Fig. 9. Average time comparison for decryption

The integrated benchmarks distinctly underscore the superior performance exhibited by the hybrid cryptographic approach. Specifically, it manifests a remarkable 20% acceleration in encryption and a substantial 16% increase in decryption efficiency in contrast to the widely utilized AES-256-bit Algorithm. These outcomes underscore the robustness and efficiency of the proposed approach,

positioning it as an innovative and pragmatic solution for fortifying content security within IPFS and decentralized storage systems at large. The hybrid cryptographic approach demonstrated notable advantages over the traditional AES-256 Algorithm in terms of speed and efficiency. The flexibility of combining symmetric and asymmetric cryptography provides an optimal balance between performance and security. Moreover, the experiment underscores the significance of addressing content security within IPFS to fortify its position as a reliable and secure decentralized storage solution.

8. Conclusion

In conclusion, this research introduces a pioneering hybrid cryptographic framework to address the critical content security gap within the IPFS. By seamlessly integrating the efficiency of symmetric key cryptography (AES 128-bit) with the robustness of asymmetric key cryptography (ECC), our proposed algorithm achieves a substantial performance boost. The benchmarks affirm a remarkable 20% acceleration in encryption speed and a 16% increase in decryption efficiency compared to the standard AES 256-bit method. This work not only advances the understanding of secure decentralized storage but also provides a practical and efficient solution to fortify content security in IPFS. The hybrid cryptographic approach emerges as a promising tool to strengthen the security fabric of blockchain-based applications reliant on decentralized storage.

References

1. Routray, S., et al. Secure Storage of Electronic Medical Records (EMR) on Interplanetary File System (IPFS) Using Cloud Storage and Blockchain Ecosystem. – In: Proc. of 4th International Conference on Electrical, Computer and Communication Technologies (ICECCT'21), November 2021, pp. 1-9.
2. J a y a b a l a n, J., et al. Blockchain and IPFS Integrated Framework for Secure EHR Management. – IEEE Trans. Biomed. Eng., Vol. **68**, 2021, No 10, pp. 2902-2910.
3. L i, J., et al. Privacy-Preserving Blockchain Model for Supply Chain Traceability. – IEEE Trans. Ind. Inform., Vol. **17**, 2022, No 8, pp. 5236-5245.
4. M e n g, W., et al. BSDNFilter: A Trust-Based Security Mechanism for Blockchain-Based SDN. – In: Proc. of IEEE Int. Conf. BlockchainCryptocurrency (ICBC'21), 2021.
5. Z h a n g, Y., et al. CPVPA: Certificateless Public Verification Scheme Using Blockchain. – IEEE Trans. Cloud Comput., Vol. **10**, 2022, No 5, pp. 994-1007.
6. Z h a n g, G., et al. Blockchain-Based Decentralized SCM System with Secure Information Sharing. – IEEE Trans. Eng. Manage., Vol. **69**, 2022, No 2, pp. 234-246.
7. H e, B., et al. Charitable Donation System Based on Ethereum Blockchain. – IEEE Trans. Serv. Comput., Vol. **15**, 2022, No 4, pp. 639-651.
8. B a r g e r, A., et al. Blockchain-Based Charity Foundation Platform on Hyperledger Fabric. – IEEE Trans. Technol. Soc., Vol. **10**, 2023, No 3, pp. 543-555.
9. U d d i n, M. N., et al. Design Principles Based on Information Security in Blockchain and IPFS. – IEEE Trans. Depends. Secure Comput., Vol. **19**, 2022, No 1, pp. 76-89.
10. D o a n, T. V., V. B a j p a i, Y. P s a r a s, J. O t t.Towards Decentralized Cloud Storage with IPFS: Opportunities, Challenges, and Future Directions. – arXiv 2022, arXiv:2202.06315.
11. K a n g, P., W. Y a n g, J. Z h e n g. Blockchain Private File Storage-Sharing Method Based on IPFS. – Sensors, Vol. **22**, 2022, No 14, 5100.

12. B a t t a h, A. A., et al. Blockchain-Based Multi-Party Authorization for Accessing IPFS Encrypted Data. – IEEE Access, Vol. **8**, pp. 196813-196825.
13. Z h a n g, Q., Z. Z h a o. Distributed Storage Scheme for Encryption Speech Data Based on Blockchain and IPFS. – The Journal of Supercomputing, Vol. **79**, 2023, No 1, pp. 897-923.
14. U l l a, M. M., M. S. K h a n, D. S. S a k k a r i. Implementation of Elliptic Curve Cryptosystem with Bitcoin Curves on SECP256k1, NIST256p, NIST521p, and LLL. – Journal of ICT Standardization, Vol. **11**, 2023, No 4, pp. 329-353.
15. B o s, J. W., et al. Elliptic Curve Cryptography in Practice. – In: Proc. of 18th International Conference of Financial Cryptography and Data Security (FC'14), Christ Church, Barbados, 3-7 March 2014, Berlin, Heidelberg, Springer, Revised Selected Papers 18, pp. 157-175.
16. S t o r u b l e v t c e v, N. Cryptography in Blockchain. – In: Proc. of 19th International Conference of Computational Science and Its Applications (ICCSA'19), Saint Petersburg, Russia, 1-4 July 2019, Springer International Publishing, Proceedings, Part II 19, 2019, pp. 495-508.
17. C h o i, S., et al. A Server-Based Distributed Storage Using Secret Sharing with AES-256 for Lightweight Safety Restoration. – IEICE Transactions on Information and Systems, Vol. **103**, No 7, pp. 1647-1659.
18. R a j p u t, S., J. S. D h o b i, L. J. G a d h a v i. Enhancing Data Security Using AES Encryption Algorithm in Cloud Computing. – In: Proc. of 1st International Conference on Information and Communication Technology for Intelligent Systems, Springer International Publishing, Vol. **2**, 2016, pp. 135-143.
19. S a s s a n i, B. A., et al. Evaluating Encryption Algorithms for Sensitive Data Using Different Storage Devices. – Scientific Programming, 2020, pp. 1-9.
20. A l z o u b i, Y e h i a I b r a h i m, A l j a a f r e h, A l i. Blockchain-Fog Computing Integration Applications: A Systematic Review. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 1, pp. 3-37.
21. J a d h a v, S. V., S. P. P a t i l, S. B. P a t i l, D. D. P a t o d i a, A. P o k h a r k a r. Decentralized Data Storage Solutions Using Hyperledger Fabric. – In: Proc. of International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT'23), Karaikal, India, 2023, pp. 1-6. DOI: 10.1109/IConSCEPT57958.2023.10170327.
22. K a n s h i, A v a n e e s h, S o u n d r a p a n d i y a n, R a j k u m a r, V. S. A n i t a S o f i a, V. R. R a j a s e k a r. Hybridized Cryptographic Encryption and Decryption Using Advanced Encryption Standard and Data Encryption Standard. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 4, pp. 63-78.

Received: 31.01.2024; Accepted: 05.03.2024 (fast track)