FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 8 No. 6, December, (Special Issue) 2024, pp 42 - 49
DOI: https://doi.org/10.33003/fjs-2024-0806-2832

ADVANCED ENCRYPTION STANDARD (AES) IMPLEMENTATION EFFICIENCY USING JAVA AND NODE.JS PLATFORMS

***[1]Ugwunna, C. O., [2]Okimba, P. E., [3]Alabi, O. A., [2]Orji, E. E., [3]Olowofeso, E. O. and [3]Ayomide, S. O.**

[1]Department of Computer Science, Wigwe University Isiokpo, River State, Nigeria.
[2]Department of Computer Science, Nnamdi Azikiwe Universirty Awka, Anambra State, Nigeria.
[3]Department of Computer Science, Federal University of Agriculture Abeokuta. Ogun State

*Corresponding authors' email: charles.ugwunna@wigweuniversity.edu.ng Phone: +2348037921011

ABSTRACT

The rapid advancement of communication technologies, such as satellite networks, mobile, internet, and terrestrial communications, has created an urgent need to protect sensitive data from potential attacks. This is particularly crucial as photos transmitted through unreliable channels may contain sensitive or confidential information. This study evaluates the effectiveness of the Advanced Encryption Standard (AES) algorithm implemented in Java and Node.js, focusing on their performance in data encryption and decryption. The research employs AES in Cipher Block Chaining (CBC) mode, using 128-bit keys for Java and 256-bit keys for Node.js. It utilizes the Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE) to create an optimized runtime environment with advanced cryptographic libraries. The result indicate that Java's AES-128 implementation is more efficient than Node.js's AES-256, particularly in terms of speed and data processing capabilities as seen in figure 11 taking Java 2.00ns to encrypt and decrypt before the Node.js algorithm that couldn't complete the process but remain at 0.75ns. Suggesting that specific use case and requirements should be considered when choosing between the two platforms for AES encryption. Java generally outperforms Node.js in efficiency, but Node.js provides essential cryptographic functions through its built-in 'crypto' module. Overall, the research underscores the advantages of using the AES algorithm across these platforms while demonstrating the varying performance characteristics between them.

Keywords: Encryption, Decryption, Advanced Encryption Standard (AES), Cryptography

INTRODUCTION

Because of the internet and communication technologies rapid development, people now share information through digital photographs. A digital image is made up of a group of matrix arranged pixels. This growing use of technologies for information storage and sharing, such as personal computers, mobile phones, and many other means of alternative communication, has increased the number of users which has increased the rate of document hacking (Anwarul & Agarwal, 2017). Due to this rapid advancement of communication technologies, such as satellite networks, mobile, internet, and terrestrial communications, there is an urgent need to thwart copying and tracking as well as to protect vital individual, general, and universal devices and their acquired data from intruders using AES (Wadi & Zainal, 2014). In addition, more unauthorized users are making an effort to collect unauthorized information. Unfortunately, by its nature, the Internet cannot guarantee the safety of the information it transmits (Goldberg & Wagner, 1996).

Knowledge is communicated or kept in an encrypted fashion to address this flaw. An unauthorized user cannot decipher this encrypted information. Information is protected both during storage and transmission. Thanks to cryptography, a branch of knowledge security science. Each secret writing algorithm and secret writing technique includes two components: an algorithm and a secret writing key usage. In order to stop the leakage of sensitive data, data encryption has become crucial for enterprises. This means that no personal employee information, including usernames, passwords, and contacts, can be transferred. Because of this, encryption is crucial to every information system. Use of a variety of coding techniques has become necessary to safeguard sensitive information from unauthorized users (Auyporn&Vongpradhip, 2015).

It is crucial to encrypt any multimedia content that needs to be transferred. Therefore, image encryption, a branch of the science of cryptography, is essential for protecting the images sent and received over mobile communications, pay-TV, e-commerce, personal emails, the transmission of financial information, the security of ATM cards, laptop passwords, and other aspects of daily life. Algorithms are used in ciphering the process of cryptography.

Kumari, Gupta and Sardana (2017) pointed out that one of the well-known methods for protecting image secrecy over a trustworthy, unconstrained public medium is image encryption. Since the medium of communication is attackable, effective encryption algorithms are a requirement for secure data transit.

According to its creators, the concurrent, class-based, object-oriented Java programming language is so basic that any developer can pick it up quickly. It is intended for widespread use. Although the Java programming language is comparable to C and C++ in function, it is structured differently, incorporating ideas from other languages while excluding some C and C++ features. Many times, Java programs are converted into bytecode that may be run on any Java Virtual Machine (JVM), regardless of the underlying computer architecture. Java contains fewer low-level objects than C and C++, but its syntax is identical to both. Dynamic functionality (such thinking and changing code while it is running) that is not often available in conventionally translated languages is provided by the Java runtime environment.

In this work, a Java cryptographic library from Gunnsteinsson (2016) was used. Free, cross-platform, and capable of running on a V8 system, Node.js is a back-end JavaScript runtime engine. Outside of a web browser, JavaScript code is executed by it. Node.js thus represents the "everywhere" paradigm, which unifies the development of web applications around a single programming language as opposed to numerous serverside and client-side scripting languages. Despite the fact that the default file extension for JavaScript code is, the name "Node.js" is only used to indicate a product in this context and does not associate with a specific file. These design strategies aim to improve the bandwidth and scalability of real-time web applications as well as web applications with various I/O activities (Node.js "Crypto | Node.js Documentation, 2021).

This study made use of the crypto library in (Nita & Mihailescu, 2022). This study encrypted data that mimics the most typical file types. The Advanced Encryption Standard, or AES as often referred to, is the appropriate algorithm. Java and node.js are two computer languages that are used to encrypt and decrypt data. In order to compare performance, the average encryption and decryption times for Java and Node.js are calculated individually. Three alternative key sizes of 128, 196, and 256 bits are used by the symmetric cryptographic method known as the Advanced Encryption Standard (AES). All initial data, including the encryption key, is calculated in bytes since the AES algorithm processes all data and encryption in bytes. 128 bits are used for the initial block size, and everything is mathematical. The length of the key determines how many times the AES algorithm will be executed. The strength of the encryption is influenced by the length of the key, the length of the key's code, and the difficulty of obtaining the key.

An incredibly contemporary Hill (AdvHill) cipher algorithm that codes using an unconscious key matrix is projected in (Acharya et al, 2009). They mishandled both the original Hill cipher algorithm and their proposed AdvHill cipher formula to encrypt entirely distinct photos. Additionally, it is obvious that the original Hill Cipher cannot correctly decrypt images that contain a lot of identically colored or grayscale space. They are generating a self invertible matrix for the Hill Cipher algorithm. Using this key matrix, they encrypted grayscale as well as colour images. Their algorithm works well for all types of gray scale as well as colour images except for the images with background of same gray level or same colour.

Zhang et al (2009) does research on algorithmic methods for image secret writing and DES confidential/secret writing as well as disorderly confidential/secret writing. The new secret writing theme's first strategy involves creating a pseudo-random sequence using a logistic chaos sequencer, carrying on the RGB with this order to the image carelessly, and then creating double time encryptions with advanced DES. Their findings demonstrate strong initial sensitivity, high security, and rapid secret writing.

Seyed et al (2010) investigated a novel algorithm for image coding supported SHA-512 hash performance. The algorithmic rule consists of 2 main sections: the primary will pre the operation to shuffle one half image. The second uses a hash function to come up with a random variety mask. The mask is then XORed with the opposite part of the image that is going to be encrypted.

A new image encryption technique based on random pixel permutation is also proposed by Indrakanti and Avadhani (2011) with the goal of preserving image quality. The method divides the encryption procedure into three steps. The picture encryption stage is the first. The crucial generation phase is the second stage. The procedure of identification is the third stage. With fewer computations, this offers anonymity to colored images. The permutation procedure is much more efficient and rapid.

Several studies have been conducted to analyze the security and performance of the AES algorithm. These studies have focused on various aspects of the algorithm, including its key size, block size, and encryption/decryption speed.

One study by Kumar et al (2020) examined the security of the AES algorithm and concluded that it provides a high level of security against various types of attacks, including differential and linear cryptanalysis. The authors also highlighted the importance of using a sufficiently long key length to ensure the security of the algorithm.

Zahid et al (2017) analyzed the Advanced Encrypting Standard (AES), and in their image encryption techniques they build on a key stream generator (A5/1, W7) to AES to confirm rising the encoding performance. A new permutation approach called RijnDael is introduced by (Bani & Jantan, 2008). It combines image permutation and acknowledgment coding scheme. The original image was divided into blocks of four pixels by four pixels, which were then rearranged using a permutation method to create a permuted image. The resultant image was then encrypted using the RijnDael formula. Their findings demonstrated that using the mix technique significantly reduced the connection between image components and increased entropy.

Nag et al (2011)outlines a two-section encoding and decryption process that works by randomly rearranging the pixels in the image using an affine redesign, then encrypting the resulting image using an XOR operation. They use affine remodel techniques with four 8-bit keys to redistribute the element values to completely other locations. The original image is then split into blocks of two-by-two pixels, with each block being encrypted using an improper XOR operation with four 8-bit keys. The method uses a sixty-four-bit full key size. Their findings showed that the correlation between element values was significantly reduced following the affine redesign.

Another study by Alenezi et al (2020) compared the performance of the AES algorithm with other encryption algorithms, such as DES and Triple DES. The authors found that the AES algorithm offered better encryption and decryption speeds, making it more suitable for applications that require high-speed data processing.

Advanced Encryption Standard (AES) implementation efficiency using Java and Node.js platforms is to investigate and analyze the performance and effectiveness of AES encryption algorithms when implemented in both Java and Node.js programming languages. This study aims to identify the strengths and weaknesses of each platform in terms of speed, memory usage, and overall efficiency when encrypting and decrypting data using the AES algorithm. By comparing the performance of AES implementation in Java and Node.js, this study will provide valuable insights into the optimal choice of platform for secure data encryption in real-world applications.

Overall, the literature on the AES algorithm demonstrates its strong security features and efficient performance, making it a popular choice for encryption in a wide range of applications. Further research is needed to continue evaluating the algorithm's security and performance in different scenarios and to explore potential improvements or enhancements to the algorithm.

MATERIALS AND METHODS

Advanced Encryption Standard (AES) is a widely used encryption algorithm that ensures secure communication and data protection. Implementing AES efficiently using Java and Node.js platforms require a systematic methodology to achieve optimal performance. The suggested methodology for implementing AES efficiently on Java and Node.js platforms involve the following steps:Understanding the AES algorithm: Before implementing AES, it is crucial to have a clear understanding of how the algorithm works and its various components such as key expansion, substitution, permutation, and mixing operations.Choosing the right AES mode: AES supports different modes of operation. For this study, the CBC was used based on the specific requirements of the application for achieving the desired efficiency.Optimizing key generation and management: Generating secure and random encryption keys, as well as implementing proper key management techniques, was crucial for ensuring the security and efficiency of the AES implementation.Leveraging hardware acceleration: Utilizing hardware acceleration features available in modern processors significantly improved the performance of AES encryption and decryption operations.Implementing parallel processing: Taking advantage of multi-threading and parallel processing capabilities in Java and Node.js platforms helped optimize the performance of AES encryption and decryption operations.Testing and benchmarking: Thoroughly testing the AES implementation on both Java and Node.js platforms and benchmarking its performance against established standards can help identify any bottlenecks and areas for further optimization.

By following these steps and adopting a systematic approach, it is possible to ensure the efficient implementation of AES using Java and Node.js platforms, thereby enhancing the security and performance of their applications.

For the purpose of gathering data for this study, we use an experimental approach. To automate and generate research data, a simple program for encryption and decryption has been developed. The obtained data was investigated using quantitative analytical methods, and the results are presented in the study's results section. This study's experiment uses symmetric keys of the AES algorithm to encryption and decrypt digital images to be communicated. AES keys of 128 bits, and 256 bits were respectively used for Java and Node.js platforms., reading each bit of the image as it is encrypted. The sidebar or an alphanumeric character (+,*), for example, can be used to generate this key. The key, known as a non-byte encrypt, will be generated using Java and Node.js crypto tools. The symmetric key must first be generated in order to create the image before encrypting the file with it. Each bit from the image is sent to the system, which then encrypts each bit before adding it to the main file. After encryption, the file will be sent to the recipient. The main goal of encryption is to prevent unauthorized access to the material. The recipient receives the file once it has been encrypted by giving them the symmetric key. The same procedure is used for decryption: the encrypted file is passed to the system, which then decrypts every bit surrounding it to create a new file with the same extension and finally outputs the results. The majority of platforms provide microsecond accuracy, but Java supports nanosecond accuracy, such that the results have the same resolution, microseconds, and precision for all systems. A total of 30 files, with 145 bytes and 32 MB apiece, were created. To ensure exceptional accuracy, the process of encrypting and decrypting this data was repeated 100 times for each combination of data size and AES key size. The experiment utilized the same hardware and software.

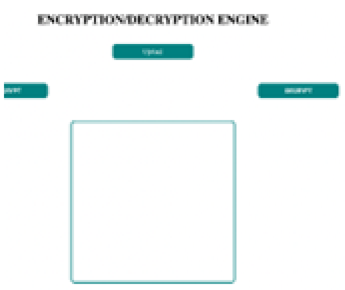
Algorithm

Making a symmetric key, which will be used to encrypt the image of the numbers. This is the sole key that will be understood by both the sender and the recipient, and it is used to create files containing encrypted data by changing the data or bit in those files to a new bit that no one else can access or understand.

```
Algorithm: Pseudocode for AES encryption algorithm
STEP 1:  Function AES (byte in[16]), byte out [16], key_array round key (Nr +1)
STEP 2:  byte state [16]
STEP 3:  state = in
STEP 4:  AddRoundKey(state,round_key[0])
STEP 5:  fori =1to Nr-1 do
STEP 6:  SubBytes(state)
STEP 7:  ShiftRows(state)
STEP 8:  Mixcolum(state)
STEP 9:  AddRoundKey(state,round_key[i])
STEP10: SubBytes(state)
STEP11: ShiftRows(state)
STEP12:  AddRoundKey(state,round_key[Nr])
STEP 13:  out = state
STEP 14:  return out
```

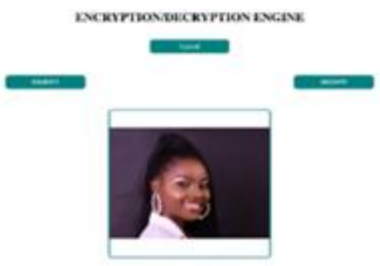
Implementation

The digital image was encrypted and decrypted using the AES technique, JavaScript (node.js) as the main programming language, and Java and Node.js as the runtime environment. Given that Java and the node.js library are efficient and quick and can quickly scale out to handle larger loads, they were used for the encryption and decryption processes. Since it can handle more demanding tasks like handling enormous amounts of data encryption and sending responses over the internet, it is the programming language of choice for creating this system. The framework of nod.js is based on Angular, while the frontend language is composed of Typescript and Javascript. Javascript serves as the backend language, while Node.JS serves as the backend framework for NodeJS. JavaScript code can be run outside of a web browser using Node.js, an open source, cross-platform runtime engine that runs on a V8 computer. Figures 1A through Figure 7G illustrate the process for encrypting and decrypting the data picture used in this research.



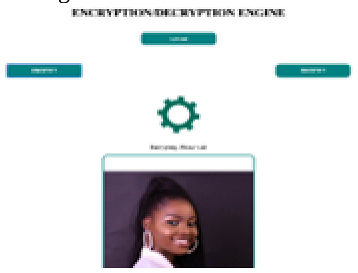
(A)

Figure 1 First interface



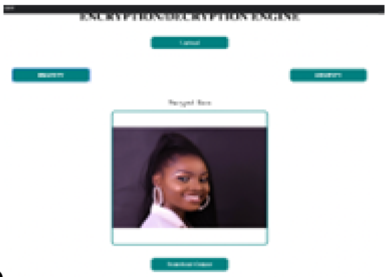
(B)

Figure 2: Upload original image



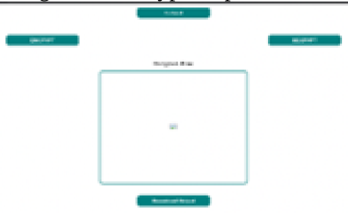
(C)

Figure 3: Encryption process



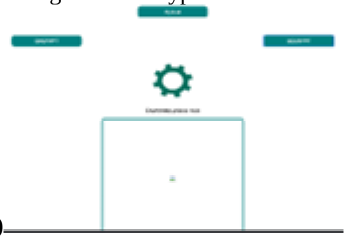
(D)

Figure 4: Encryption Done



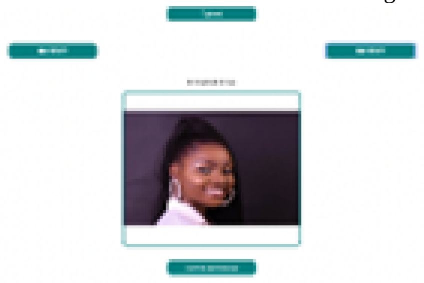
(E)

Figure 5: Upload encrypted image to decrypt



(F)

Figure 6: Decryption process



(G)

Figure 7: Decryption Done

RESULTS AND DISCUSSION

Once the Java and Node.js programs that generate the study data depicted in table 1 and the included information therein have been executed, it generates the dataset arising from this analysis as shown in tables 2 and 3. Figure 11 illustrates the typical encryption and decryption times per iteration. The cumulative results of 100 repetitions determine the time required for data encryption and decryption. By raising the number of retries, you gain weight by lowering the noise that may be caused by unplanned operating system events. By dividing these figures by the overall size of the encrypted data, the total time per Byte for each combination of key size, programming language, and platform is computed. The mean time (in ns) for complete encryption and decryption, based on the platform, operation, and key size, shows that Java AES implementations are currently the most effective and fast in terms of performance using the AES128 algorithm, surpassing node.js with the AES256 algorithm. The typical times, in milliseconds, needed to perform encryption and decryption operations on each content platform inside the zone of 145 bytes at 32M are shown in Figs. 9 and 12. Based on the normal time required to fully decrypt the data, Fig. 11 provides a visual conclusion that, in terms of performance, Java has generally demonstrated to be the best option for data encryption in the algorithm tested even at 128-bits AES computation. The mean time (in ns) for full encryption and decryption demonstrates that Java AES implementations are now the most efficient and quick in terms of performance, exceeding node.js. According to the amount of the encrypted data, the key size, which is determined by the key length, in bytes, and the overall encryption and decryption time, a group of raw data records can be formed for further analysis. Fig. 13 displays data on the typical encryption and decryption times in nanoseconds for a byte of data. Analysis of the information gathered was also used to obtain the data.

Table 1: Experimental result

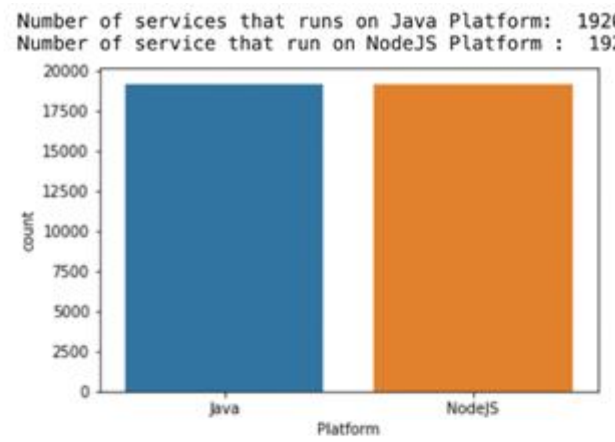
	Key Size	Payload Size	Duration (ns)
count	38400.000000	3.84000e+04	3.84000e+04
mean	192.000000	2.211935e+06	2.786460e+06
std	52.256462	6.161608e+06	8.243116e+06
min	128.000000	1.450000e+02	4.919000e+03
25%	128.000000	9.815000e+02	1.856875e+04
50%	192.000000	2.540000e+05	3.465215e+05
75%	256.000000	9.182500e+05	1.567990e+06
max	256.000000	3.2000000e+07	1.996791e+08

Table 2: First Set of Data

Key Size	Payload Size	Duration (ns)	Operation	Platform
128	145	172091	ENCRYPTION	Java
128	145	293555	DECRYPTION	Java
128	145	193368	ENCRYPTION	Java
128	145	142647	DECRYPTION	Java
128	145	136840	ENCRYPTION	Java
128	145	3269115	DECRYPTION	Java
128	145	175830	ENCRYPTION	Java
128	145	135301	DECRYPTION	Java
128	145	139426	ENCRYPTION	Java
128	145	128157	DECRYPTION	Java
128	145	147642	ENCRYPTION	Java

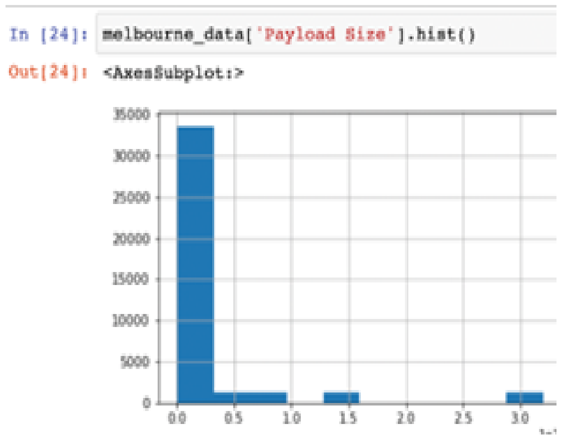
Table 3: Last Set of Data

Count	Key Size	Payload Size	Duration (ns)	Operation	Platform
38370	256	32000000	35080951	ENCRYPTION	NodeJS
38371	256	32000000	21711575	DECRYPTION	NodeJS
38372	256	32000000	36095070	ENCRYPTION	NodeJS
38373	256	32000000	13664608	DECRYPTION	NodeJS
38374	256	32000000	35010851	ENCRYPTION	NodeJS
38375	256	32000000	19336487	DECRYPTION	NodeJS
38376	256	32000000	36217529	ENCRYPTION	NodeJS
38377	256	32000000	20392842	DECRYPTION	NodeJS
38378	256	32000000	36473128	ENCRYPTION	NodeJS
38379	256	32000000	19944245	DECRYPTION	NodeJS



```
Number of services that runs on Java Platform:  1920
Number of service that run on NodeJS Platform :  192
```

Figure 8: Platform (Java & Node.js)



```
In [24]: melbourne_data['Payload Size'].hist()
Out[24]: <AxesSubplot:>
```

Figure 9: Platform (Java & Node.js)

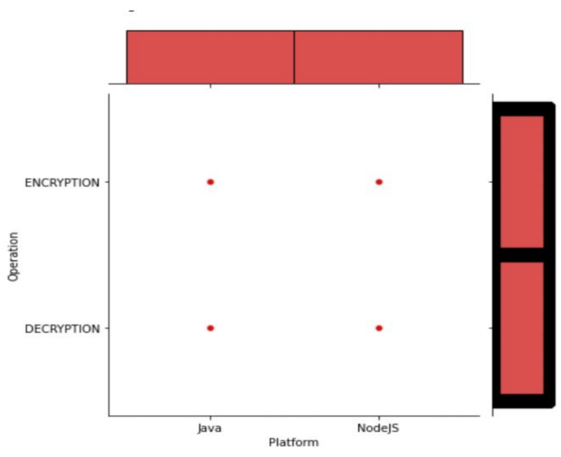


Figure 10: Operation (Encryption & Decryption)

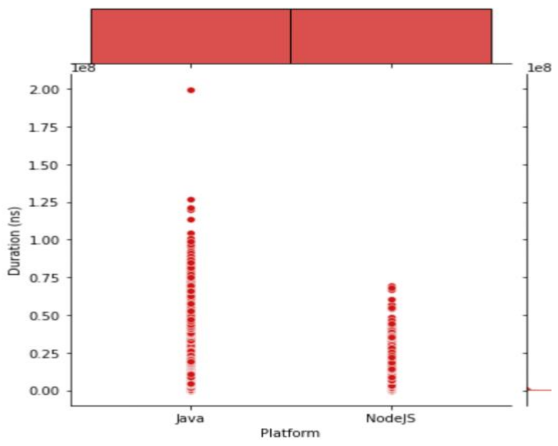


Figure 11: Duration (ns)

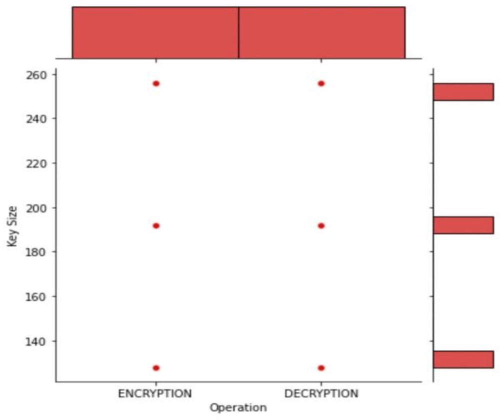


Figure 12: Key Sizes (128, 192, 256)bits

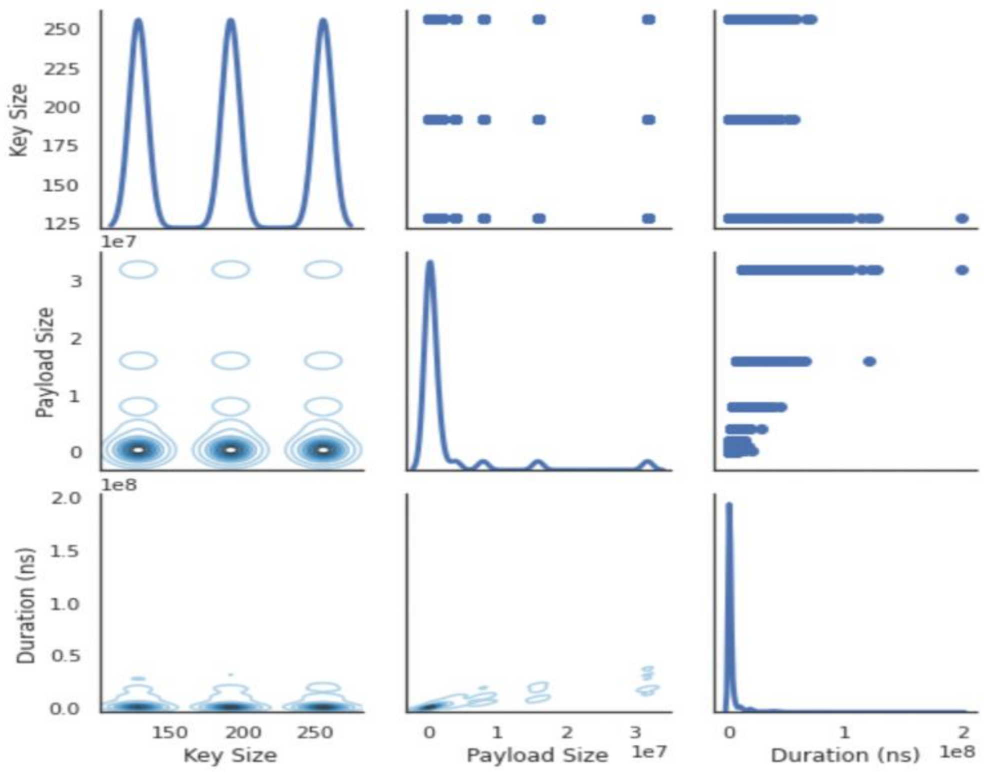


Figure 13: Experimental Result

Based on the findings, comparing the performance of AES implementation on Java and Node.js platforms, several factors come into play that influenced efficiency differences.

In Java, AES encryption was noticed to be typically faster due to its strong emphasis on performance optimization and efficient memory management. Java's Just-In-Time (JIT) compiler allows for code to be dynamically compiled and optimized, resulting in faster execution times. Additionally, Java's multithreading capabilities enable parallel processing, further enhancing performance.

On the other hand, Node.js, being a JavaScript runtime built on Google's V8 engine, may not perform as well when it comes to AES encryption. JavaScript is inherently single-threaded, which can limit the ability to leverage multiple cores for parallel processing. This can result in slower execution times compared to Java.

Furthermore, the underlying libraries and implementations used in each platform can also impact performance. Java's standard cryptographic libraries are well-established and optimized for performance, whereas Node.js may rely on external libraries that may not be as efficient.

Benefits

The following are the benefits of using either the Java or Node.js platforms in CBC picture encryption using AES128 and AES256 algorithms respectively;

Data security: The use of AES algorithm in digital image (picture) encryption ensures that the data is securely encrypted, making it difficult for unauthorized users to access the information.

Privacy protection: By encrypting pictures using cipher block chain and AES algorithm, the privacy of the individuals in the pictures is protected, as the encrypted data can only be accessed by authorized users.

Tamper-proofing: The use of cipher block chain ensures that the encrypted pictures are tamper-proof, as any attempt to alter the data would be immediately detected.

Data integrity: The AES algorithm ensures that the integrity of the encrypted pictures is maintained, as any unauthorized changes to the data would be detected and prevented.

Secure sharing: The use of cipher block chain and AES algorithm allows for secure sharing of encrypted pictures, as only authorized users with the correct decryption keys can access the information.

Compliance with regulations: Encrypting pictures using AES algorithm and cipher block chain helps organizations comply with data protection regulations and standards, as it ensures the security and privacy of sensitive information.

CONCLUSION

In conclusion, data encryption and decryption using the AES algorithm in CBC mode with 128 bits, and 256-bit keys were tested. Experience has also shown that when it comes to online platforms, the speed of data encryption and decryption using Java programming exceeds node.js even with lesser bits. We can draw the conclusion that Java, as opposed to node.js, is the platform with the best effective implementation of AES. On the other hand, the node.js AES implementation's performance fell short of expectations with higher bits as compared to Java platform. Node.js took longer than Java to conduct the encryption and decryption operations while utilizing a 128-bit key. AES encryption can be implemented in web platforms using both Java and Node.js, but each has unique advantages and limitations. Java is a good option for computationally difficult workloads because of its performance, numerous cryptography libraries, and mature environment. On the other hand, Node.js excels at asynchronous I/O operations and offers a more straightforward development and deployment environment. The specific needs of your project, such as performance requirements, available libraries, and the general architecture of your application, will determine which of the two platforms you should use.

REFERENCES

Acharya, B., Panigrahy, S. K., Patra, S. K., & Panda, G. (2009). Image encryption using advanced hill cipher algorithm. *International Journal of Recent Trends in Engineering*, *1*(1), 663-667.

Alenezi, M. N., Alabdulrazzaq, H., & Mohammad, N. Q. (2020). Symmetric encryption algorithms: Review and evaluation study. *International Journal of Communication Networks and Information Security*, *12*(2), 256-272.

Anwarul, S., & Agarwal, S. (2017). Image enciphering using modified AES with secure key transmission. In Communication and Computing Systems: Proceedings of the International Conference on Communication and Computing Systems (ICCCS 2016), Gurgaon, India, 9-11 September, 2016 (p. 137). CRC Press.

Auyporn, W., &Vongpradhip, S. (2015). A robust image encryption method based on bit plane decomposition and multiple chaotic maps. *Int. J. Signal Process. Syst*, *3*(1),

Bani, M. A., & Jantan, A. (2008). Image encryption using block-based transformation algorithm. *IJCSNS International Journal of Computer Science and Network Security*, *8*(4), 191-197.

Goldberg, I., & Wagner, D. (1996). Randomness and the Netscape browser. *Dr Dobb's Journal Software Tools for the Professional Programmer*, *21*(1), 66-71.

Gunnsteinsson, O. (2016). A search for a convenient data encryption algorithm-For an Internet of Things device 8-13.
Indrakanti, S. P., &Avadhani, P. S. (2011). Permutation based image encryption technique. *International Journal of Computer Applications*, *28*(8), 45-47.

Kumari, M., Gupta, S. & Sardana, P. (2017) A Survey of Image Encryption Algorithms. *3D Res* **8**, 37. https://doi.org/10.1007/s13319-017-0148-5

Kumar, K., Ramkumar, K. R., & Kaur, A. (2020, June). A design implementation and comparative analysis of advanced encryption standard (AES) algorithm on FPGA. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 182-185). IEEE.

Nag, A., Singh, J. P., Khan, S., Ghosh, S., Biswas, S., Sarkar, D., & Sarkar, P. P. (2011). Image encryption using affine transform and XOR operation. In *2011 International conference on signal processing, communication, computing and networking technologies* (pp. 309-312). IEEE.

Nita, S. L., & Mihailescu, M. I. (2022). Java Cryptography Architecture. In Cryptography and Cryptanalysis in Java: Creating and Programming Advanced Algorithms with Java SE 17 LTS and Jakarta EE 10 (pp. 29- 46). Berkeley, CA: Apress.

Node.js, "Crypto | Node.js Documentation. Available: https://nodejs.org/api/cryptohtml # crypto. [Accessed 4 April 2021].

Seyed Mohammad Seyedzade, Reza Ebrahimi Atani and Sattar Mirzakuchaki, A Novel Image Encryption Algorithm Based on Hash Function 6th Iranian Conference on Machine Vision and Image Processing, 2010

Wadi, S. M., & Zainal, N. (2014). High-definition image encryption algorithm based on AES modification. *Wireless personal communications*, *79*, 811-829.

Zahid,M., Machhout,M., Khriji,L., Baganne,A., Tourkianalyze,R., (2017) "Advanced encoding Standard (AES)"

Zhang,Y., Liu,W., Cao, S., Zhai, Z., Nie,X., and Dai,W.,(2009). "Secret writing & speed sensitivity