**Comparative Analysis of the Speeds of AES, ChaCha20, and Blowfish Encryption/Decryption Algorithms**

**Maninder Kaur1, Nicholas Fishel1**

1 Hamilton Southeastern High School, Fishers, Indiana

**Student Authors**

Maninder Kaur - High School

**SUMMARY**

With the increasing reliance on digital data and sensitive information, the need for robust data protection methods became paramount. The research commenced with an examination of the pressing problem of data breaches and unauthorized access that continued to plague modern information systems as a form of background context. This research paper delved into the realm of encryption algorithms and their effectiveness in safeguarding data in a fast-paced manner. The study's core objective was to compare and contrast the speed of three prominent encryption algorithms: Advanced Encryption Standard (AES), ChaCha20, and Blowfish. To address this question, the main hypothesis tested if ChaCha20 encryption algorithm would enable the faster encryption and decryption among the three. To obtain results, the researchers employed a set of trials using lorem ipsum text files of various lengths. Each algorithm was subjected to rigorous testing to assess its encryption and decryption times. While each algorithm demonstrated unique strengths, a clear winner, ChaCha20, emerged concerning speed. These results offered valuable context to decision-makers seeking an optimal encryption solution for the specific need of speedy encryption, and this study's value lay in its contribution towards understanding a suitable encryption algorithm that could address this challenge effectively.

**INTRODUCTION**

In an era where our online data was so very important, ensuring its security against unauthorized users became a major purpose. There were 1,802 large-scale reported data compromises last year (1), leading to numerous individuals having their most valuable pieces of information exposed to unauthorized hands. One such security that could be implemented to protect data was encryption and decryption algorithms. Defined as a method of converting data from plaintext (unencrypted) to ciphertext (encrypted); These algorithms played a pivotal role in safeguarding sensitive information from malicious intent (2). The significance of encryption and decryption algorithms lay in their ability to render plaintext data incomprehensible to anyone without the appropriate decryption key. Such encryption algorithms play a crucial role in safeguarding sensitive information across various types of data files (text documents, images, videos, audio files, executables, and more).

Yet, each file type has a difference of which it is encrypted, even though they all follow the same process. When one opens a file’s contents with a terminal window, the majority of PDF, DOCX, etc. files will represent readable words. On the other hand, images, videos, and other non-document file types are depicted as binary in the form of gibberish symbols with the terminal. Understanding the pre-encrypted content allows for readability of the file content. However, after encryption, both files become unreadable gibberish blobs of symbols. On top of that, encryption algorithms for each file type varies based on platform. Adobe Acrobat is a common PDF and document type encrypted, yet unable to encrypt others. The same can be said for video, image, and more (3,4). While the plaintext content of a PDF file may differ from that of an MP4 file (3), encryption algorithms transform these files into ciphertext, solidifying a wall of security between attackers with intent to harm (4). This process ensures that even if attackers gain access to encrypted data, they cannot discern its original meaning without the appropriate decryption keys. Thus, encryption techniques create a robust security barrier against malicious intent, irrespective of the type of data being protected (2,5). This algorithm ensured that only people with the correct cryptographic key will ever be able to decrypt the encrypted files. As data volumes grew exponentially, the demand for efficient encryption techniques intensified, making the speed of encryption and decryption a critical factor in evaluating their suitability for real-world applications. The primary objective of this study was to conduct a comparative analysis of three widely adopted encryption and decryption algorithms: Advanced Encryption Standard (AES), ChaCha20, and Blowfish. The researchers aimed to answer the hypothesis: Did the ChaCha20 encryption and decryption algorithm outperform AES and Blowfish in terms of speed? Our investigation aimed to shed light on the significance of speed in data protection, ultimately unveiling which algorithm proved most adept at safeguarding data from unauthorized access without compromising on efficiency.

Three encryption algorithms were selected for this study - AES (Advanced Encryption Standard), ChaCha20, and Blowfish, all of which were symmetric encryption algorithms, instead of asymmetric (4). Symmetric and Asymmetric encryption were two different forms of encryption. Symmetric encryption used the same key to both encrypt and decrypt a file, while an asymmetric encryption algorithm utilized different keys for the encryption and decryption process. The reason the researchers only tested symmetric encryption was that they were normally faster in encryption and decryption, as the computer did not have to take too much time accessing two different keys; the computer simply used one (5, 6).

The bit sizes varied due to different levels of performance for each bit. To understand this, it's essential to understand the inner mechanics of each of the algorithms. ChaCha20 normally comes in a 256 key size (7). Blowfish, due to it being the oldest out of all the algorithms, is normally used within a robust 128 key size (7). Lastly, AES is the one algorithm with more variety in key sizes, often these being 128 and 256 (7). ChaCha20 with a 256-bit key offers a comparable level of security to AES with a 128-bit key (7, 8). This ensures that the comparisons are fair in terms of security strength. ChaCha20 is designed to be efficient in software implementation, making it competitive in speed against AES 128 and Blowfish 128 (7, 8, 9). By evaluating ChaCha20 256 against AES 128 and Blowfish 128, the researchers can understand the performance trade-offs between newer and older algorithms, as well as between different cipher designs discussed later (stream vs. block).

Bits were the “words” of the computer world. Computers understood commands through a collection of 1s and 0s. For example, the 4-bit sequence of 1000 would mean the number 8. A key, or what allowed encrypted data, to be reversed to the original plain text, consisted of a sequence of bits. Without the key, an encrypted piece of data could not be decrypted to its original form. So, in this study when the researchers stated that they would be utilizing AES 128-bit, that meant that the key consisted of 128 bits that allowed encrypting/decrypting the chosen file. The bigger the bit, normally the stronger and more complex the encryption (7).

Despite all the algorithms used being symmetric encryption algorithms, AES, Blowfish and ChaCha20 also all have distinct characteristics that influence their performance within different scenarios. The choice between these algorithms often depends on the specific requirements of the application. AES-128's hardware support provides a performance advantage in many scenarios, while ChaCha20's software efficiency makes it preferable in contexts where hardware acceleration is unavailable. Blowfish, though flexible with its key length, is generally considered less secure for large-scale data encryption due to its smaller block size. Understanding these differences helps in selecting the most appropriate encryption algorithm for a given use case.(8, 9, 10)

One of the encryption algorithms the research team tested was the Advanced Encryption Standard (AES), a symmetric-key encryption algorithm with key sizes of 128, 192, or 256 bits. Within the research process, the researcher utilized AES with a key size of 128, as its an industry standard (4, 10). AES had been developed as a replacement for the aging Data Encryption Standard (DES). AES was adopted by the U.S. National Institute of Standards and Technology (NIST) in 2001 as a federal government standard. AES employed block cipher encryption, dividing plaintext into fixed-size blocks and applying multiple rounds of transformation to produce ciphertext (10).

Its selection had been based on its robustness, speed, and suitability for a wide range of applications. Its robust security can be showcased with its multiple key sizes (128, 192, and 256), providing different levels of security based on the different situations. The key sizes provide possible keys. For example, an AES 128-bit key size has the ability to produce 2^128 possible keys. This is another reasons as to why AES 128 was chosen, as the larger the key size becomes, the more complex and longer the process may take (4, 10). Among global standards, it's acclaimed for these security measures (4, 10). Regrading its speed, AES can achieve encryption speeds of several gigabytes per second, hitting standard operational benchmark goals to today’s encryption algorithms (4, 10, 11). With these characteristics in mind, the real world applications range from basic TLS/SSL internet security protocols, to storage protection, to secure messaging on social media (12, 13, 14).

ChaCha20 is a stream cipher and symmetric encryption algorithm designed to provide high security and fast performance with a fixed key size of 256 bits. A stream cipher is a process of encrypting a file by pushing every binary digit in the data through the algorithm via a cryptographic key (15). It was created by Daniel J. Bernstein in 2008 as part of the eSTREAM project. ChaCha20 is known for its simplicity and ease of implementation, while still offering a strong level of security against various cryptographic attacks (7). Its design aimed to achieve a good balance between security, speed, and resistance to side-channel attacks (EXPLAIN). ChaCha20 is widely used in various applications, including secure communication protocols, disk encryption, and digital signature schemes (16, 17).

Blowfish is a symmetric-key block cipher encryption algorithm designed by Bruce Schneier in 1993 that uses variable key sizes, with the recommended key size being 128 bits. Block ciphers differ from stream ciphers as instead of encrypting every bit of data, the cipher puts data into blocks and encrypts them as groups (18). It was developed as an alternative to the aging Data Encryption Standard (DES) and provides a more secure and efficient solution for encrypting sensitive data. Blowfish operated on 64-bit blocks of plaintext and supported variable key lengths, making it adaptable to different security requirements. One of the key strengths of Blowfish is its simplicity, which allows for a relatively easy implementation and fast encryption and decryption processes (19). Additionally, Blowfish has undergone extensive cryptanalysis and has proven to be resilient against various attacks, further enhancing its reputation as a reliable encryption algorithm.

While newer algorithms like AES and ChaCha20 gained popularity in recent years, Blowfish remained a noteworthy historical milestone in the field of cryptography and continued to be used in various applications where speed and security were essential (20). The three encryption algorithms were chosen specifically since they shared multiple characteristics. They were all symmetric encryption algorithms, were widely used, and had different key sizes to choose from.

Several research studies examined the security and speed aspects of different encryption algorithms.

A research experiment conducted a comparative analysis of multiple encryption algorithms, AES and Blowfish being two of them. In the study, the team assessed the algorithms' performance in terms of multiple factors (architecture, flexibility, reliability, security, and limitation) to known cryptographic attacks. This study provided insight on efficiency also, which roughly translates and gives a basic foundation of how fast certain algorithms go. Allowed insight into the fact that AES was relatively faster than Blowfish, Blowfish could have limitations if key size is smaller, and that both are still widely used today. (21). Their findings shed light on the strengths and weaknesses of each algorithm, contributing to the understanding of their real-world applicability.

Similarly, another paper delved into an analysis of the ChaCha20 cipher and its impact on the development of modern cryptographic techniques. Due to the fact that the previous didn’t have ChaCha20 to be tested, this paper was used to grow larger knowledge into this algorithm. Their exploration of the algorithm's mathematical foundations provided valuable context for understanding its strengths in asymmetric encryption (22). The research paper dwelled into the security aspects of the encryption algorithm, along with key sizes and other differences that evaluate the performance. The key sizes was an important discussion within the paper, as it explores how it differs with how long it takes to encrypt/decrypt (22).

Lastly, another research paper observed the performance of encryption algorithms in the real world with IoT (Internet of Things) devices (23). All three encryption algorithms that are being studied in this research paper are also used within the IoT devices research paper, finally allowing inferences and comparisons from the stream and block ciphers (23). This comparison came in to help later on in the understanding of just how different configurations of key sizes are needed to be equal to one another (23).

Between the three research papers used as previous studies, there were multiple similarities between the first two and the third research paper. Key sizes comparisons, explanations over stream vs block cipher, explanations over performance measures are just some of anime that were commonalities (21, 22, 23). Another commonalities between all three research papers is that none talk enough about the speed (21, 22, 23). “Performance” doesn't refer to anything regarding time spent, only vaguely mentioned occasionally. All research papers focus on security and limitations, rather than observing which is the fastest (21, 22, 23). This is a gap that this research paper is addressing.

Our hypothesis theorized that the ChaCha20 algorithm would have a faster encryption and decryption time compared to the AES and Blowfish. In this paper, the researchers were focused on finding which encryption and decryption algorithm provided fast service: AES, ChaCha20, or Blowfish? In the real world, different length of files are encrypted. For all these varying scenarios, simply doing one file length wouldn’t suffice. Thus, testing different file types was essential to answering the research question that applies in the real world. By running multiple different text files of varying lengths multiple times through both algorithms, and recording the average of each trial, researchers were able to conclude which algorithm upheld the fastest encryption/decryption sequence. Findings within the research conducted found that ChaCha20 does indeed work faster than both AES and Blowfish, even by just a little. Unlike the other two algorithms who had larger margins of comparison between the encryption and decryption times, ChaCha20 showcased more balanced times. On top of this, testing different text files found little to no difference, as there was no discernible difference or jumps in time as text file sizes increased. More key findings are discussed and examined later in the paper. The efficiency of an encryption/decryption algorithm majorly lay within how fast the data would run through. The purpose of this paper was to allow individuals to find which encryption algorithm better suited their situation relating to text file encryption. (ADD MORE SOURCES)

**RESULTS**

The study compared the performance of three encryption algorithms—AES, ChaCha20, and Blowfish—across various text file lengths: sentence, paragraph, page, chapter, section, and book. Each algorithm underwent testing for encryption and decryption speed through ten trials for each text file length. The lengths of each file were as follows (total number of characters, approximated): sentence: 15-20, paragraph: 150-200, page: 300-700, chapter: 3000-5000, section: 10000-20000, and book: 100000.

The average encryption and decryption times, measured in seconds, are summarized in Table 1, derived from the time taken for each encryption and decryption process for each text file length over 10 trials. All conclusions drawn stated below are inferred and concluded from the data in Table 1.

AES consistently exhibited a shorter average decryption time than ChaCha20, but longer than Blowfish across all text file lengths. However, its decryption time was generally faster than ChaCha20 and Blowfish. Notably, AES decryption was significantly faster than AES encryption, emphasizing the decryption simply includes deciphering text with a key, while encryption includes creating a key and then enciphering the plaintext (4, 10, 15).

Blowfish generally performed comparably to AES in terms of encryption time, but consistently had longer decryption times. It was notably slower than ChaCha20 for both encryption and decryption.

ChaCha20 consistently outperformed AES and Blowfish in terms of encryption and decryption speed for all text file lengths. When averaging both operations, ChaCha20 had the shortest average time (0.00372 seconds) compared to AES (0.0973 seconds) and Blowfish (0.00892 seconds) (data from Table 1). The algorithm's encryption and decryption times were relatively balanced and showed consistent performance across different lengths.

AES exhibited better performance in decryption compared to its encryption times, while ChaCha20 maintained a more balanced performance in both operations.

Blowfish, being an older encryption algorithm, demonstrated slower performance compared to the other two modern algorithms in all cases.

Despite the varying sizes of the text files, there was no discernible pattern indicating an increase or decrease in encryption/decryption times. All three algorithms consistently completed their processes in approximately the same amount of time for each of the data files.

**DISCUSSION**

From the experiment, we aimed to compare the speed of three encryption algorithms—AES, ChaCha20, and Blowfish—when applied to text files of various lengths. Our findings reveal that ChaCha20 emerged as the fastest encryption algorithm, showcasing consistently faster encryption and decryption times compared to AES and Blowfish. While AES demonstrated competitive decryption times, it generally lagged behind in encryption speed, whereas Blowfish proved slower overall. These results offer valuable insights for researchers and developers seeking optimal encryption solutions for different applications.

A quick clarification would go a long way. The encryption process includes both encryption and decryption, yet encryption and decryption are two different things.

Readers may have noted how similar times were for each file, regardless of size, when observing each algorithm’s performance from table 1.

There are multiple unique features of ChaCha20 that allowed the algorithm to provide the best performance regarding speed out of the three algorithms tested. The stream cipher mechanisms discussed in the third research appear (23), point towards the fact that stream ciphers are in general more simplistic yet still efficient in properly performing quick cryptographic transformations (15, 18, 23). Such mechanic include the inner boolean operations, as the algorithm specifically uses addition, rotation, and xor operations (different computational arithmetic computers used to solve problems) for calculations and transformations (15, 16, 17). These operations are computationally parallelized to work quicker than the other two algorithms, who have much more complex operations (16, 17). The most prominent feature that adds uniqueness is the differences in hardware integrations. For AES and Blowfish, both are in need of hardware-accelerated hardware (5, 10, 19). In other words, a device that is fast. On the other hand, ChaCha20, works regardless of hardware capabilities (16, 17). This could easily change the results of the research experiment, if the experiment was not performed on a MacBook, but on an older PC or on a modern gaming PC.

Another question that needs to be answered is why exactly encryption is decryption times for algorithms either differ or stay almost consistent. AES’s encryption and decryption is very asymmetric, as according to Table 1. This is due to the differences in the key schedule process for encryption and decryption. During encryption, the key expansion, or the “schedule”, is straightforward, whereas decryption involves an inverse key schedule, which can be slightly more computationally difficult (4). This mechanic causes decryption for both AES and Blowfish to be asymmetric. ChaCha20 also had asymmetry, yet not as large as the other two algorithms, thus it is not considered alongside the others.

As observed from Table 1, Blowfish was the slowest out of all three of the algorithms. The reasons lie with the older algorithms' mechanics. The key schedule for Blowfish is far more complex, once source even stating it to be “notably complex” (4, 19, 20). This is due to the older algorithms operations (4, 19, 20). Newer algorithms often replace entire operations with one operation symbol process. Blowfish, however, does not, making it longer to encrypt/decrypt thorough its complexity (19, 20). This same difference in age can also be pointed towards how hardware capabilities aren’t properly integrated. Blowfish was created in the 1990s, and due to it being far earlier than the other algorithms, modern-day hardware has trouble with integration, making any algorithmic process strenuous and slower (4, 20).

Another observation that needs to be discussed is how time for encryption and decryption does not change based on how large the file is. Researchers initially hypothesized that the larger the file, the slower the process might be. Thus, the trials would grow by each file size. There were two main reasons as top why this was an incorrect hypothesis. One, the hardware acceleration of the device was capable enough to perform cryptographic operations in real time (3, 2, 26). This bridges into the second reason, steam and block ciphers mechanically are able to split data, regardless of size, into fixed number sets that are transformed (4, 18). Block ciphers create a certain number of blocks and fit data into them to encrypt, regardless of size (18). Stream ciphers, as mentioned before, encrypt bit by bit (15, 18). While the process is considered more strenuous, the stream ciphers often use hardware capabilities to encrypt bits faster (15, 18). This is why there was no clear increase in time as file size grew.

Something that have come to notice is the use of seconds to portray how long the encryption and decryption process took. This is an ideal approach rather than choosing time per character, which has some faults. For one, every algorithm initializes prior to encrypting (4). This could easily skew results for shorter initialization processes, and should not be considered alongside actual encryption time per character (4). As mentioned above, there was no real pattern for the increase or decrease of characters. Thus, it would be wrong to compare time per character if the algorithm’s goal is to encrypt/decrypt in a fixed time due to uniform character processing.

Yet, there are possible limitations and drawbacks in each encryption algorithm and with our experiment. ChaCha20 is known for its efficiency in software implementations, especially on devices without hardware support for AES instructions (4, 5, 10). In some cases, ChaCha20 with a 256-bit key can provide comparable or better performance than AES128, making it an attractive choice for applications where computational efficiency is critical (4, 11, 16, 17). However, due to ChaCha20 being a relatively newer algorithm, it might not have undergone the same level of scrutiny as AES (14). On the topic of AES, Its weaknesses include potentially slower performance in certain applications that don’t have hardware-acceleration, mobile devices, software-only situations, and susceptibility to timing attacks (4, 10, 11). Blowfish is known for its slower performance compared to modern algorithms, and the absence of widespread adoption in critical applications limit its use in scenarios where speed is a priority (4, 19, 23). The experiment's limitations include the use of pseudocode instead of actual code execution, which might not account for all implementation details and optimizations in specific programming languages and libraries. Additionally, the experiment focused solely on speed performance and did not assess the algorithms' resistance to advanced attacks. Real-world implementations also consider other factors, such as memory usage and platform support, which were not addressed in this study.

It is also essential to acknowledge some limitations of the experiment process. First, the experiments were conducted on a single machine, and results may vary on different hardware configurations. A computer operating on different hardware accelerations, RAM, GPUs, etc. may produce different results as it's been discussed how each algorithm responds differently to different hardware capabilities (3, 6, 8).

Furthermore, the language used was English, thus not completely representative of all languages that these encryption algorithms can encrypt.

Also, the length of each text file is corresponding with a certain average length (such as a paragraph composed of 6 sentences, a page of 3 paragraphs, etc.) which may not be totally factual. One book may have 700 pages while another has 50. A paragraph can have 15 sentences, or 3 sentences. In an attempt to rectify this error in our experiment, we generated text files with average lengths of their corresponding writing elements.

Lastly, the code and pseudocode was used for simulation purposes, and actual implementation may have subtle differences. The code that was used and written was from an encryption library. It was done on some code files. Yet, often times, these algorithms are integrated within a software, and that encrypts files separately. The real world implication would be much more complex, while the research purpose code was miniscule to the amount of complexity in the real world.

Our research could have other aspects taken into consideration while also assisting in other research opportunities. For example, future research should consider evaluating the algorithms' security strengths and weaknesses with malware or viruses, including resistance to more advanced attacks. Such advanced attacks include side-channel attacks or attacks that analyze time for cryptographic operations, analyzing electromagnetic emissions from devices, and power consumption of cryptographic operations to reveal information of the key or the text (25).

Asymmetric systems are also better than symmetric cryptographic systems. All research done used symmetric encryption algorithms. Symmetric is faster, but in the context of more secure algorithms, asymmetric wins (4). For future study, for those that require more security, testing only asymmetric systems would provide more context in this field.

Another consideration is investigating the algorithms' performance in resource-constrained environments, such as Internet of Things (IoT) devices or embedded systems, could provide valuable insights for practical applications like smart home devices, wearable technology, and automotive systems to name a few (26).

On top of all of this, previous research utilized certain encryption algorithms that this research experiment did not utilize. This research focuses on AES, ChaCha20, and Blowfish. For future research, other research groups could utilize these algorithms alongside others not tested. There are reasons as to why some weren’t tested. One of the biggest reasons was that all-tested algorithms were symmetric algorithms, while asymmetric algorithms like RSA, ECC, DSA were not tested (4). Testing similar algorithms was the goal, and straying away by picking asymmetric would not be ideal. Another reason was the key sizes. Lower or larger key sizes can cause slower encryption/decryption time, but also not be around the ideal standard of 128-bit key sizes, like the algorithms RSA, DES/3DES, and ECC that have key sizes not around this ideal key size (4).

Alongside this, one could also explore hybrid encryption approaches, combining the strengths of different algorithms, may offer improved security and performance for specific use cases. Lastly, extending the experimentation to include real-world implementations in various programming languages and platforms, such as Python, Java, C++, and embedded systems like Arduino or Raspberry Pi, can validate the findings and identify potential optimization opportunities (27, 28). If an individual or group wants to improve or add onto our research, they should consider testing different types of data files, more algorithms, and different hardware settings. A video file is far more complex than a text file, thus there is a possibility that a certain algorithm may work better for such file types unlike others.

This study sheds light on the critical role of encryption algorithms in safeguarding digital data and highlights the importance of considering speed in algorithm selection. By identifying ChaCha20 as a standout performer in terms of speed and efficiency, this research offers a promising avenue for enhancing data security speed in various domains. Moving forward, continued exploration of encryption techniques and their real-world applications will undoubtedly drive advancements in cybersecurity, ensuring a safer digital landscape for individuals and organizations alike.

**MATERIALS AND METHODS**

The experiments were conducted on a computer with a standard configuration, including an Intel Core i5 processor, 8 GB RAM, and a solid-state drive (SSD). The operating system used was a macOS Monterey Version 12.6. The experiments were performed using Python 3.x with the `cryptography` library. All Python libraries were updated and ran on the latest versions (All compatible with Python 3.11, thus the cryptography library was running on version 41.0.1 at the time). Code and pseudocode were developed for each encryption algorithm to simulate the encryption and decryption processes (29). The pseudocode for AES, ChaCha20, and Blowfish was adapted from standard Python implementations using the `cryptography` library. Each algorithm was tested ten times on each of the text files, with the size of the file ranging from a few bytes to a few kilobytes.

The experiment utilized five distinct text files with diverse lengths of Lorem Ipsum content as the input data for encryption and decryption. The content was created using a Lorem Ipsum generator (30), and was used as Lorem Ipsum can mimic real life data well. The text files were of varying lengths, comprising a sentence, a paragraph, a page, a chapter, a section, and the length of a complete book.The text files were preprocessed to ensure consistent formatting and content. The lengths of each file were as follows (total number of characters, approximated): sentence: 20, paragraph: 200, page: 500, chapter: 5000, section: 20000, and book: 100000.

We used the following key sizes to ensure equivalency between all algorithms: AES 128, ChaCha20 256, and Blowfish 128 bits. The reason the ChaCha20 had a larger bit size was in part to differences in the design and structure of the algorithms. While AES 256 and Blowfish 256 have a larger key size, the security of an encryption algorithm depends on various factors, including key schedule, block size, and the algorithm's resistance to known attacks.Thus, ChaCha20 256 was chosen to be equal in comparison with AES and Blowfish in terms of performance ability as the mechanics for the key size of ChaCha20 256 is similar to the lower key sizes of 128 that AES and Blowfish possess in the experiment (4, 6, 9, 10). Thus, no encryption algorithm has an advantage over another,

The collected data was analyzed to compare the average encryption and decryption times for each algorithm. The results were tabulated to visualize the relative performance of AES, ChaCha20, and Blowfish across different text file lengths.

**REFERENCES**

1. ‘Identity Theft Resource Center’s 2022 Annual Data Breach Report Reveals near-Record Number of Compromises’. *ITRC*, Identity Theft Resource Center, 25 Jan. 2023, https://www.idtheftcenter.org/post/2022-annual-data-breach-report-reveals-near-record-number-compromises/.
2. *What Is Encryption? | IBM*. www.ibm.com/topics/encryption.
3. Sruthi, S., et al. “Encryption and decryption of text file and audio using LabVIEW.” *ResearchGate*, July 2017, https://doi.org/10.1109/netact.2017.8076816.
4. *Handbook of Applied Cryptography*. cacr.uwaterloo.ca/hac.
5. *Encryption Choices: Rsa vs. Aes Explained*. 15 June 2021, https://preyproject.com/blog/types-of-encryption-symmetric-or-asymmetric-rsa-or-aes.
6. “Performance Analysis of Encryption Algorithms for Security.” *IEEE Conference Publication | IEEE Xplore*, 1 Oct. 2016, ieeexplore.ieee.org/document/7955835.
7. ‘All about Cryptographic Bit Lengths’. *TCC - Technical Communications Corporation: Network Encryption, Secure Communications*, https://www.tccsecure.com/NewsResources/CipherONEBlog/TabId/1222/ArtMID/1578/ArticleID/2/All-About-Cryptographic-Bit-Lengths.aspx. Accessed 29 May 2024.
8. Al Tamimi, Abdel-Karim. *Performance Analysis of Data Encryption Algorithms*. www.cs.wustl.edu/~jain/cse567-06/ftp/encryption\_perf.
9. Bernstein, Daniel J. “ChaCha20, a variant of Salsa20.” *University of Chicago*, Department of Mathematics, Statistics, and Computer Science, chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://cr.yp.to/chacha/chacha-20080120.pdf
10. https://www.nist.gov/publications/advanced-encryption-standard-aes. Accessed 29 May 2024.https://www.nist.gov/publications/advanced-encryption-standard-aes
11. Daemen, Joan, and Vincent Rijmen. “The Design of Rijndael.” *Information security and cryptography*, 2002, https://doi.org/10.1007/978-3-662-04722-4.
12. “SSL And TLS: Designing and Building Secure Systems 0201615983, 9780201615982 - EBIN.PUB.” *ebin.pub*, ebin.pub/ssl-and-tls-designing-and-building-secure-systems-0201615983-9780201615982.html.
13. Paolomatarazzo. “BitLocker Overview - Windows Security.” *Microsoft Learn*, 6 Nov. 2023, learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker.
14. “Apple Platform Security.” *Apple Support*, support.apple.com/guide/security/welcome/web.
15. ‘Stream Cipher’. *Security*, TechTarget, 23 Apr. 2021, https://www.techtarget.com/searchsecurity/definition/stream-cipher.
16. Computerphile. *Chacha Cipher - Computerphile*. Youtube, 19 Feb. 2021, https://www.youtube.com/watch?v=UeIpq-C-GSA.
17. Nir, Yoav, and Adam Langley. ‘RFC 7539: ChaCha20 and Poly1305 for IETF Protocols’. *IETF Datatracker*, https://datatracker.ietf.org/doc/html/rfc7539. Accessed 29 May 2024.
18. ‘Difference between Block Cipher and Stream Cipher’. *GeeksforGeeks*, 20 May 2019, https://www.geeksforgeeks.org/difference-between-block-cipher-and-stream-cipher/.https://www.geeksforgeeks.org/difference-between-block-cipher-and-stream-cipher/
19. ‘Blowfish Algorithm with Examples’. *GeeksforGeeks*, 14 Oct. 2019, https://www.geeksforgeeks.org/blowfish-algorithm-with-examples/.
20. Awati, Rahul. “Blowfish.” *Security*, 19 Jan. 2022, www.techtarget.com/searchsecurity/definition/Blowfish.
21. *Kumari, Shailja, and Jyoti Chawla. Comparative Analysis on Different Parameters of Encryption Algorithms for Information Security. May 2015. - Google Search*. https://www.google.com/search?q=Kumari,+Shailja,+and+Jyoti+Chawla.+Comparative+Analysis+on+Different+Parameters+of+Encryption+Algorithms+for+Information+Security.+May+2015.&oq=Kumari,+Shailja,+and+Jyoti+Chawla.+Comparative+Analysis+on+Different+Parameters+of+Encryption+Algorithms+for+Information+Security.+May+2015.&gs\_lcrp=EgZjaHJvbWUyBggAEEUYOdIBBzQxNWowajeoAgCwAgA&sourceid=chrome&ie=UTF-8. Accessed 29 May 2024.
22. Shi et al.. “Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha.” *Lecture notes in computer science*, 2013, pp. 337–51. https://doi.org/10.1007/978-3-642-37682-5\_24.
23. Singh, Praneet, and Kedar Deshpande. “Performance Evaluation of Cryptographic Ciphers on IoT Devices.” *arXiv.org*, 5 Dec. 2018, arxiv.org/abs/1812.02220.
24. ‘Cybersecurity of Quantum Computing: A New Frontier’. *SEI Blog*, https://insights.sei.cmu.edu/blog/cybersecurity-of-quantum-computing-a-new-frontier/. Accessed 29 May 2024.
25. “What Is a Side-Channel Attack? How It Works.” *GeeksforGeeks*, 31 Mar. 2024, www.geeksforgeeks.org/what-is-a-side-channel.
26. Kamal R., “Securing the IoT Data Landscape: IoT Encryption Algorithms - Intuz.” *Intuz*, www.intuz.com/blog/securing-the-iot-data-landscape-iot-encryption-algorithms.
27. Team, Codecademy. “7 Best Programming Languages for Cryptography.” *Codecademy Blog*, 9 Apr. 2024, www.codecademy.com/resources/blog/programming-languages-for-cryptography.
28. Daniel, Brett. “What Are Embedded Systems?” *Trenton Systems*, 18 Mar. 2024, www.trentonsystems.com/en-us/resource-hub/blog/what-are-embedded-systems.
29. Maninder Kaur. “GitHub - Themaninderkaur/Research-paper-sources: Code, Pseudocode and Text Files Used in Research Paper.” *GitHub*, github.com/themaninderkaur/research-paper-sources.
30. “Lorem Ipsum – Generator, Origins and Meaning.” *Lorem Ipsum*, loremipsum.io.

**Tables with Captions**

Table 1: Average time in seconds of each encryption algorithm to encrypt/decrypt a text file containing a certain length of Lorem Ipsum. The data was collected by recording the time taken for each encryption and decryption process for each text file length over 10 trials, and the mean was calculated resulting in these values.

| Length | AES Encryption | AES Decryption | ChaCha20 Encryption | ChaCha20 Decryption | Blowfish Encryption | Blowfish Decryption |
| --- | --- | --- | --- | --- | --- | --- |
| Sentence | 0..01024 | 0.0006 | 0.0028 | 0.0009 | 0.097 | 0.001 |
| Paragraph | 0.099 | 0.0007 | 0.0028 | 0.0011 | 0.098 | 0.0013 |
| Page | 0.099 | 0.0007 | 0.0026 | 0.0009 | 0.09 | 0.0012 |
| Chapter | 0.09 | 0.0008 | 0.0029 | 0.0011 | 0.083 | 0.0017 |
| Section | 0.092 | 0.0008 | 0.0027 | 0.0011 | 0.097 | 0.0011 |
| Book | 0.086 | 0.0002 | 0.0027 | 0.0007 | 0.086 | 0.0011 |