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

**Maninder Kaur1, Nicholas Fishel1**

1 Hamilton Southeastern High School, Fishers, Indiana

**Student Authors**

Maninder Kaur - High School

**SUMMARY**

With our increasing reliance on digital data and sensitive information, there is a paramount need for robust data protection methods. Our study examined encryption algorithms and their effectiveness in safeguarding data in a fast-paced manner. Our core objective was to compare the speed of three prominent encryption algorithms: Advanced Encryption Standard (AES), ChaCha20, and Blowfish. We hypothesized that the ChaCha20 encryption algorithm would enable the fastest encryption and decryption among the three. To test this, we conducted 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. For each file length text file, each of the encryption algorithms would be tested on the file 10 consecutive times and the average was accessed. While each algorithm demonstrated unique strengths (such as faster encryption time, faster decryption times, similar encryption/decryption times, easier implementation of pushing files to be encrypted into the algorithm, ease of use/coding, etc.), ChaCha20 emerged as a clear winner. These results offer valuable context to decision-makers in sectors in need of implementing security for their data (large technological corporations data, academic institutions’ student information, medical facilities’ patient charts, etc.) seeking a rapid encryption solution.

**INTRODUCTION**

In the digital era, it is vital to ensure the security of our online data against unauthorized users. There were 1,802 global large-scale data breaches (breaches that affect a large amount of people, compromises extremely sensitive data, and/or tremendously impacts the victims in a negative way) reported last year , leading to numerous individuals having their most valuable information exposed (1). One security measure that could be implemented to protect data is encryption algorithms. Defined as methods of converting data from plaintext (unencrypted) to ciphertext (encrypted), these algorithms play a pivotal role in safeguarding sensitive information from malicious intent (2). Encryption algorithms 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, including text documents, images, videos, audio files, and executables.

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).

Three encryption algorithms were selected for this study: Advanced Encryption Standard (AES), ChaCha20, and Blowfish, all of which are symmetric encryption algorithms (4). Symmetric encryption uses the same key to both encrypt and decrypt a file, while asymmetric encryption algorithms utilize different keys for the encryption and decryption processes. We only tested symmetric encryption algorithms because they are generally faster in encryption and decryption (5, 6).

A “bit” is a unit of computational information (7, 8). Computers understand 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 allows encrypted data to be reversed to the original plain text, consists of a sequence of bits. Without the key, an encrypted piece of data cannot 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 larger the number of bits, normally the stronger and more complex the encryption (8). The key sizes provide possible keys. For example, an AES 128-bit key size can produce 2^128 possible keys.

Despite all being symmetric encryption algorithms, AES, Blowfish, and ChaCha20 have distinct characteristics that influence their performance within different scenarios. 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 (4-6). Hardware support and acceleration enhance performance by offloading cryptographic operations like the algorithms above to specialized hardware, which is faster and more efficient than software-based methods (8,9). Blowfish, though flexible with its key length, is generally considered less secure for large-scale data encryption due to its smaller block size (5). The block size represents the amount of data processed in one go during encryption or decryption; Smaller block sizes, like Blowfish's 64-bit, are less secure for large data due to the increased likelihood of repeating patterns, which can be exploited in attacks. Larger block sizes, like AES's 128-bit, mitigate these risks (5,8,9). Understanding these differences helps in selecting the most appropriate encryption algorithm for a given use case.

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. . Like AES, Blowfish 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 operates on 64-bit blocks of plaintext and supports 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 (10). Additionally, Blowfish has undergone extensive cryptanalysis and has proven to be resilient against various attacks, further enhancing its reputation as a reliable encryption algorithm (11,12). While newer algorithms like AES and ChaCha20 gained popularity in recent years, Blowfish remains a noteworthy historical milestone in the field of cryptography and continues to be used in various applications where speed and security are essential (11).

Another one of the encryption algorithms we tested was AES, a symmetric-key encryption algorithm with key sizes of 128, 192, or 256 bits. In this study, the researcher utilized AES with a key size of 128, as it is an industry standard since it provides security and performance for majority security problems (4, 13). AES was developed as a replacement for the aging Data Encryption Standard (DES) and was adopted by the U.S. National Institute of Standards and Technology (NIST) in 2001 as a federal government standard. AES employs block cipher encryption, dividing plaintext into fixed-size blocks and applying multiple rounds of transformation to produce ciphertext (13). AES was selected by NIST based on its robustness, speed, and suitability for a wide range of applications. The selection of AES was made by the U.S. National Institute of Standards and Technology (NIST) as part of the process to replace the aging Data Encryption Standard (DES). Its versatile security levels can be showcased with its multiple key sizes (128, 192, and 256), providing different levels of security based on the different situations. This is another reason why AES 128 was chosen for testing in this manuscript alongside the other algorithms, as the larger the key size becomes, the longer the encryption and decryption processes may take (4, 14). AES can achieve encryption speeds of several gigabytes per second, hitting standard operational benchmark goals for modern encryption algorithms (4, 13, 14). With these characteristics in mind, the real-world applications range from basic TLS(Transport Layer Security)/SSL(Secure Sockets Layer) cryptographic internet security protocols for secure internet browsing, to storage protection, to secure messaging on social media (15-17).

ChaCha20 is a symmetric encryption algorithm designed to provide high security and fast performance with a fixed key size of 256 bits. ChaCha20 is also a stream cipher, meaning it encrypts files by pushing every binary digit in the data through the algorithm via a cryptographic key (18). Block ciphers differ from stream ciphers in that, instead of encrypting every bit of data, the cipher puts data into blocks and encrypts them as groups (19). It was created by Daniel J. Bernstein in 2008 as part of the eSTREAM project (2004-2008), a EU-funded initiative led by ECRYPT, aiming to create new stream ciphers as replacements for older algorithms. The project had a significant impact on cryptographic research and created multiple notable ciphers, one being ChaCha20, a variation of another cipher created at the time, Salsa20 (18). ChaCha20 is known for its simplicity and ease of implementation while still offering strong security against various cryptographic attacks (4,18). The design of ChaCha20 aimed to achieve a balance between security, speed, and resistance to side-channel attacks (attacks that exploit physical information leaked during the execution of cryptographic algorithms, rather than attacking the algorithms themselves) (19). ChaCha20 is widely used in various applications, including secure communication protocols, disk encryption, and digital signature schemes (20,21).

These three encryption algorithms were chosen for the following shared characteristics: they are all symmetric encryption algorithms, are widely used, and have different key sizes from which to choose. Several research studies examined the security and speed aspects of different encryption algorithms. A study comparing multiple encryption algorithms, including AES and Blowfish, assessed the algorithms' performance against known cryptographic attacks in terms of multiple factors, namely architecture, flexibility, reliability, security, and limitation. This study found that AES was relatively faster than Blowfish, that Blowfish could have limitations if the key size is smaller (22). One of Blowfish’s advantages is its variable key length, which ranges from 32 bits to 448 bits, allowing for greater flexibility in security levels. Blowfish is also known for its speed and efficiency, particularly in software implementations, where it can outperform AES in terms of processing speed due to its simpler structure and smaller block size of 64 bits (10-12). This leads to faster process times in applications that require high throughput. Additionally, Blowfish's design allows it to be easily implemented in resource-constrained environments, making it suitable for applications where computational efficiency is critical (4,12). These prior findings shed light on the strengths and weaknesses of each algorithm, contributing to the understanding of their real-world applicability.

Similarly, another study analyzed the ChaCha20 cipher and its impact on the development of modern cryptographic techniques. Given the newness of ChaCha20, this paper was used to build knowledge about this algorithm. This study explored the algorithm's mathematical foundations to provide valuable context for understanding its strengths in asymmetric encryption (23). The research paper analyzed security aspects of the encryption algorithm, along with key sizes and other differences that impacted performance. The key sizes were an important discussion within the paper, as it explores how it differs with how long it takes to encrypt/decrypt (23).

Still, none of these previous studies discussed speed (22, 23). Our research study aimed to address that gap. As data volumes grow exponentially, the demand for efficient encryption techniques intensifies, 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 compare three widely adopted encryption and decryption algorithms: Advanced Encryption Standard (AES), ChaCha20, and Blowfish.

We hypothesized that the ChaCha20 algorithm would have a faster encryption and decryption time compared to AES and Blowfish. In the real world, files of different lengths are encrypted. Thus, testing different file lengths was essential for the real-world applications of our research question. By running text files of varying lengths multiple times through both algorithms and recording the average of each trial, we were able to identify which algorithm was fastest for encryption and decryption. We found that ChaCha20 did indeed work faster than AES and Blowfish. Unlike the other two algorithms, which differed in their encryption and decryption times, ChaCha20 achieved more balanced times. Additionally, there was no discernible difference in required time as text file sizes increased. The significance of this paper was to allow individuals to find which encryption algorithm better suited their situation relating to text file encryption. If an individual needs the utmost speedy encryption, they can find it.

**RESULTS**

We compared the performance of three encryption algorithms, AES, ChaCha20, and Blowfish, across various text file lengths: sentence, paragraph, page, chapter, section, and book. We tested the encryption and decryption speed of each algorithm through ten trials for each text file length. The average encryption and decryption times, measured in seconds, (**Table 1**), were derived from the time taken for each encryption and decryption process for each text file length over 10 trials.

AES consistently exhibited a shorter average encryption time than Blowfish, but longer than ChaCha20 across all text file lengths. However, its average decryption time was generally faster than ChaCha20 and Blowfish. Notably, AES decryption was strikingly faster than AES encryption, reflecting that decryption simply requires deciphering text with a key, while encryption includes creating a key and then enciphering the plaintext (19,21).

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) (**Table 1**). The algorithm's encryption and decryption times were relatively balanced and showed consistent performance across different text lengths.

Despite the varying sizes of the text files, there was no discernible correlation between encryption or decryption times and file length. All three algorithms consistently completed their processes in approximately the same amount of time for each of the data files.

**DISCUSSION**

In this study, we aimed to compare the speed of three encryption algorithms, AES, ChaCha20, and Blowfish, when applied to text files of various lengths. Our findings revealed that ChaCha20 was the fastest encryption algorithm, showcasing consistently faster encryption and decryption times compared to AES and Blowfish. While AES demonstrated competitive decryption times to ChaCha20, it generally lagged behind in encryption speed, whereas Blowfish proved slower overall for both processes. These results offer valuable insights for researchers and developers seeking optimal encryption solutions. It was clear that ChaCha20 was the winner, with its on average encryption and decryption times added together to beat both Blowfish and AES.

There are multiple unique features of ChaCha20 that may have allowed the algorithm to provide the fastest performance out of the three algorithms tested. Stream ciphers are generally simpler, which may make them more efficient at performing quick cryptographic transformations (4,10). Such mechanics include the inner boolean operations, as the algorithm specifically uses addition, rotation, and xor operations (different computational arithmetic that computers used to solve problems) for calculations and transformations (18-21). These operations are computationally parallelized to work quicker than the other two algorithms, which use more complex operations (18,21). Furthermore, both AES and Blowfish need a fast device capable of hardware acceleration (12,13). On the other hand, ChaCha20 works regardless of hardware capabilities (23). We performed our experiment on a MacBook, but we note that the results may differ if repeated on an older PC or on a modern gaming PC.

Another remaining question concerns why encryption and decryption times for algorithms either differ or stay consistent. AES’s encryption and decryption are very unequal (**Table 1**). This imbalance is likely 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 unlike encryption, takes less time as there are no extra processes required (24). As a result, decryption for both AES and Blowfish was faster than encryption. ChaCha20 also had asymmetry, yet not as large as the other two algorithms.

Blowfish was the slowest of the three algorithms, likely due to the older algorithms' mechanics. The key schedule for Blowfish is “notably complex”, or that the process of generating and organizing keys is not simple (4,11). It involves multiple transformations, iterations, and bitwise operations that make it harder to reverse or predict the subkeys from the original key. Newer algorithms often replace entire operations with one operation symbol process. Since Blowfish does not do this, this algorithm takes a long to encrypt and decrypt through its complexity (4,11).

Surprisingly, we found that time for encryption and decryption did not change based on file size. Initially, we initially thought that larger files would take longer to process. We propose two potential explanations for this unexpected finding. One, the hardware acceleration of the device may have been capable of performing cryptographic operations in real time..Where devices have hardware acceleration and, consequently, can perform cryptographic operations in real time, the relationship between file size and processing time is no longer directly related. The ability to process data in fixed-size blocks, leverage parallel processing, and maintain high throughput means that larger files can be encrypted or decrypted quickly, often resulting in processing times that do not correlate directly with the total file size. This is particularly important in applications requiring fast and secure data handling. (2,3,25). This bridges into the second reason, stream and block ciphers are able to split data, regardless of size, into fixed number sets that are transformed (4). Block ciphers create a certain number of blocks regardless of the original contents of the file (10). Stream ciphers, by contrast, encrypt bit by bit . While the latter process is considered less efficient, stream ciphers often use hardware capabilities to encrypt bits faster (4, 10). Hence, there was no clear increase in encryption or decryption time as file size grew.

It has been noted that the duration of the encryption and decryption processes is often measured and reported in seconds. 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 skew results for shorter initialization processes and should not be considered alongside actual encryption time per character (4). In our experiments, we found no correlation between processing time and character number. Thus, it may not be optimal to compare time per character if the algorithm’s goal is to encrypt/decrypt in a fixed time due to uniform character processing. And naturally, this behavior is larger in bigger files. But small file trials may well show variability due to different overheads, while increasing file sizes make the general performance more consistent across board. Each algorithm has relative strengths, and the right choice may depend on usage scenarios and performance requirements.

Each encryption algorithm has potential limitations. 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 new algorithm, it might not have undergone the same level of scrutiny as AES (14). This scrutiny is from academic researchers that create and review the ciphers, government agencies like NIST, and the cryptographic industry (4,13,22). They evaluate their scrutiny based on strength, mathematical foundations, vulnerabilities, performance, and many more all of which is helpful to create ciphers that can be used and will keep data safe. The weaknesses of AES include increased susceptibility to timing attacks, as well as potentially slower performance in certain applications that lack hardware-acceleration and in software-only situations (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).

Limitations of this experiment 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 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 we did not consider in this study. Alongside this, we conducted the experiments on a single machine, and results may vary on different hardware configurations. A computer with different hardware acceleration, RAM, GPU, or other parameters may produce different results (3, 6, 9). Furthermore, we studied only text that used the Latin alphabet, which may not be completely representative of all languages. Lastly, the code and pseudocode were used for simulation purposes, and actual implementation may have subtle differences. We used code from an encryption library (26-30). We expect that real-world encryption algorithms would be much more complex. If one were to test actual encryption algorithms that are available on their everyday computers, the results may vary.

Our research could have other aspects taken into consideration that were not highlighted in this manuscript. For example, future research could evaluate the algorithms' security strength against malware or viruses, including resistance to more advanced attacks. Such advanced attacks include side-channel attacks or attacks include the ability of power consumption of cryptographic operations to reveal information about the key or the text (24). Asymmetric systems also exceed symmetric cryptographic systems in certain contexts. All experiments in this paper was done using symmetric encryption algorithms. While symmetric systems are generally faster, asymmetric algorithms are generally more secure (4). Thus, future study of 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. This could provide valuable insights for practical applications like smart home devices, wearable technology, and automotive systems (25). Additionally, future research could examine other algorithms beyond the three we 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 (4,5). 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). Future experiments may utilize these algorithms for testing. Future work could also explore hybrid encryption approaches to combine the strengths of different algorithms. It would also be important to include real-world implementations in various programming languages and platforms, such as Python, Java, C++, and embedded systems like Arduino or Raspberry Pi (31,32). Lastly, future research can consider different types of data files. A video file is far more complex than a text file, for example, so there is a possibility that a certain algorithm may work better for such file types.

We examined encryption algorithms due to their critical role in safeguarding digital data and were particularly interested in the role of speed in algorithm selection. We identified ChaCha20 as a standout performer in terms of speed for text files of various sizes. 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**

Experiments were conducted on a computer with 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 (version 41.0.1). All Python libraries were updated and run on the latest versions (all compatible with Python 3.11). Code and pseudocode were developed for each encryption algorithm to simulate the encryption and decryption processes (33). 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 used five text files with different lengths of Lorem Ipsum content as the input data for encryption and decryption. The content was created using a Lorem Ipsum generator, which can accurately mimic real-life data (34). Text files were generated of varying lengths to emulate 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, in approximate total number of characters, were: sentence: 15-20, paragraph: 150-200, page: 300-700, chapter: 3000-5000, section: 10000-20000, and book: 100000.

We used the following key sizes to ensure equivalency between all algorithms: AES, 128 bits; ChaCha20, 256 bits; and Blowfish, 128 bits. We chose a larger bit size for ChaCha20 due in part to differences in the design and structure of the algorithms. ChaCha20 256 was chosen to be comparable to AES and Blowfish in terms of performance ability (4).

**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, 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, 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. “Bit (Communications).” *Encyclopaedia Britannica*, www.britannica.com/technology/bit-communications.
8. “All About Cryptographic Bit Lengths.” *TCC - Technical Communications Corporation: Network Encryption, Secure Communications*, www.tccsecure.com/NewsResources/CipherONEBlog/TabId/1222/ArtMID/1578/ArticleID/2/All-About-Cryptographic-Bit-Lengths.aspx. Accessed 29 May 2024.
9. Al Tamimi, Abdel-Karim. *Performance Analysis of Data Encryption Algorithms*. www.cs.wustl.edu/~jain/cse567-06/ftp/encryption\_perf.
10. “Difference Between Block Cipher and Stream Cipher.” *GeeksforGeeks*, 20 May 2019, www.geeksforgeeks.org/difference-between-block-cipher-and-stream-cipher/.
11. “Blowfish Algorithm with Examples.” *GeeksforGeeks*, 14 Oct. 2019, www.geeksforgeeks.org/blowfish-algorithm-with-examples/.
12. Awati, Rahul. “Blowfish.” *Security*, 19 Jan. 2022, www.techtarget.com/searchsecurity/definition/Blowfish.
13. “Advanced Encryption Standard (AES).” *NIST*, www.nist.gov/publications/advanced-encryption-standard-aes. Accessed 29 May 2024.
14. Daemen, Joan, and Vincent Rijmen. “The Design of Rijndael.” *Information Security and Cryptography*, 2002, https://doi.org/10.1007/978-3-662-04722-4.
15. “SSL and TLS: Designing and Building Secure Systems.” *EBIN.PUB*, ebin.pub/ssl-and-tls-designing-and-building-secure-systems-0201615983-9780201615982.html.
16. Paolomatarazzo. “BitLocker Overview - Windows Security.” *Microsoft Learn*, 6 Nov. 2023, learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker.
17. “Apple Platform Security.” *Apple Support*, support.apple.com/guide/security/welcome/web.
18. 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.
19. “What Is a Side-Channel Attack? How It Works.” *GeeksforGeeks*, 31 Mar. 2024, www.geeksforgeeks.org/what-is-a-side-channel.
20. Computerphile. “ChaCha Cipher - Computerphile.” *YouTube*, 19 Feb. 2021, www.youtube.com/watch?v=UeIpq-C-GSA.
21. Nir, Yoav, and Adam Langley. “RFC 7539: ChaCha20 and Poly1305 for IETF Protocols.” *IETF Datatracker*, datatracker.ietf.org/doc/html/rfc7539. Accessed 29 May 2024.
22. Kumari, Shailja, and Jyoti Chawla. “Comparative Analysis on Different Parameters of Encryption Algorithms for Information Security.” *Proceedings of the Conference*, 2015, api.semanticscholar.org/CorpusID:212544681.
23. Shi, et al. “Improved Key Recovery Attacks on Reduced-Round Salsa20 and ChaCha.” *ICISC'12: Proceedings of the 15th International Conference on Information Security and Cryptology*, 2013, pp. 337–51. https://doi.org/10.1007/978-3-642-37682-5\_24.
24. “Cybersecurity of Quantum Computing: A New Frontier.” *SEI Blog*, insights.sei.cmu.edu/blog/cybersecurity-of-quantum-computing-a-new-frontier/. Accessed 29 May 2024.
25. Kamal, R. “Securing the IoT Data Landscape: IoT Encryption Algorithms.” *Intuz*, www.intuz.com/blog/securing-the-iot-data-landscape-iot-encryption-algorithms.
26. “Cryptography.” *Cryptography Documentation*, cryptography.io/en/latest/.
27. “os module documentation.” *Python Documentation*, https://docs.python.org/3/library/os.html.
28. “time module documentation.” *Python Documentation*, docs.python.org/3/library/time.html.
29. “Cryptographic Hashes.” *Cryptography Documentation*, cryptography.io/en/latest/hazmat/primitives/cryptographic-hashes/.
30. “Rocket Software Documentation.” docs.rocketsoftware.com/bundle/unidataunibasiccommands\_rg\_824/page/vmn1685024690247.html.
31. Team, Codecademy. “7 Best Programming Languages for Cryptography.” *Codecademy Blog*, 9 Apr. 2024, www.codecademy.com/resources/blog/programming-languages-for-cryptography.
32. Daniel, Brett. “What Are Embedded Systems?” *Trenton Systems*, 18 Mar. 2024, www.trentonsystems.com/en-us/resource-hub/blog/what-are-embedded-systems.
33. Kaur, Maninder. “GitHub - Themaninderkaur/Research-paper-sources: Code, Pseudocode and Text Files Used in Research Paper.” *GitHub*, github.com/themaninderkaur/research-paper-sources.
34. “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 text.** The data was collected by recording the time taken for each encryption and decryption process for each text file length, averaged over 10 trials.

| 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 |