Evaluation of XChaCha20-Poly1305 for Improved File System Level Encryption in the Cloud

MSc Research Project
Cyber Security

Kahol Gaurav Bhuvanagiri Udayakumar
Student ID: x20157983

School of Computing
National College of Ireland

Supervisor : Dr. Arghir-Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kahol Gaurav Bhuvanagiri Udayakumar		
Student ID:	20157983		
Programme:	MSc. In Cybersecurity	**Year:**	2021-2022
Module:	Research Project		
Supervisor:	Arghir-Nicolae Moldovan		
Submission Due Date:	16/12/2021		
Project Title:	Evaluation of XChaCha20-Poly1305 for Improved File System Level Encryption in the Cloud		
Word Count:	7893	**Page Count**	23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Kahol Gaurav Bhuvanagiri Udayakumar

Date: 15/12/2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluation of XChaCha20-Poly1305 for Improved File System Level Encryption in the Cloud

Kahol Gaurav Bhuvanagiri Udayakumar
20157983

Abstract

Cloud computing is a modern-day infrastructure for millions of applications on the Internet, and migrating to the cloud continues to pose a concern of exposing sensitive information's. Confidential information is susceptible to leak in the public domain if the storage service is misconfigured. So, the Cloud service provider (CSP) encourages organisations to enable encryption, relying primarily on the Advanced Encryption Standard (AES) to secure data. Consequently, the fundamental goal of this research is to undertake an alternative algorithm called XChaCha20 for file encryption in Cloud infrastructure and to evaluate XChaCha20 as compared to AES. The comprehensive experimental evaluation showed that XChaCha20 outperforms AES in terms of metrics such as execution time, keyspace analysis and overall throughput increased by thirty percentage while working with large files.

1 Introduction

In cybersecurity, the study of encryption is referred to as cryptography. Cryptography Research is actively carried out to discover weaknesses in current algorithms and develop a better encryption algorithm than the present algorithm. When it comes to storing or transmitting secret information, encryption is a technique used to store or transmit any type of data, including but not limited to a password, an e-book, media files, and a variety of other types of data. In computing, data decryption retrieves the original information from an encrypted source when the right key is supplied. This procedure is only possible when an individual provides the key that can be used to recover the original file.

The algorithms for encryption are classified into two types: symmetric algorithms and asymmetric algorithms. Symmetric cryptography algorithms use the same key for encryption and decryption mechanisms as mentioned by Thabit et al. [24], whereas asymmetric cryptography algorithms require both the sender's and receiver's public and private keys. Asymmetrical encryption is extensively used to transfer data across a communication channel securely. It removes the threat of sniffing a user's private key over a network, as well as the risk of eavesdropping and sniffing attacks on the data being sent. To protect sensitive information stored in databases or file systems, symmetrical encryptions can be used to encrypt passwords or Binary Large Object Files (BLOB).

An extended variant of the Chacha20 symmetric algorithm will be explored in this research. Internally, the Xchacha20 algorithm uses the Chacha20 to perform encryption on the block of bytes, while externally, this algorithm uses the HChacha20 hash function to derive subkey and sub-nonce from the actual key and extended nonce, respectively. Poly1305 is a message authentication code (MAC digest) used to verify the integrity of the original message, as shown under section 2.5 of the IEFT Protocol Manual [16]. Poly1305 uses a 32-byte secret key and a nonce to calculate a 16-byte message digest for each message block. An intruder will be unable to forge the message during an encrypted state as the sender stores an authenticator hash among each message, and the decryption validates each authenticator before processing the message.

Cloud computing is a rapidly expanding computing sector, and cloud infrastructure allows businesses to dynamically scale their computing requirements in response to those of demands from consumers, according to their organisation needs. Storage, compute, bandwidth, load balancing, virtual network management, security integrations, vulnerability with threat scanners, content delivery network (CDN), and analytical tools are just a few of the services provided by CSP for corporate infrastructure. When it comes to cloud computing, the payment strategy is "pay as you go," this implies that a company or user only pays for the services that they use, which is often less expensive for small sector enterprises. Typically, cloud service providers (CSP) such as Google Cloud Provider [1], Amazon Web Services [2], and Microsoft Azure [3] rely on symmetric-based encryption and employ AES for disk encryption and file-system level encryption of data [39]. For a key size of 256 bits, the AES method is subjected to 14 rounds of encryption, according to Adomnicai's Research [4]. In order to encrypt data, the suggested algorithm, XChaCha20, can be used, which conducts 20 rounds of encryption on 256 bits key. In general, increasing rounds makes it difficult to penetrate for cryptanalyst under Ciphertext-Only analysis and Known Plaintext analysis attack. The objective of this research is to extensively evaluate XChaCha20 against Advanced Encryption standard algorithm and report the outcomes.

1.1 Background and Motivation

Due to the Covid-19 pandemic, the vast majority of services started relying on cloud-based solutions to fulfil their requirements. For example, pharmaceutical organisations use cloud services like Object storage for storing medical prescriptions, generating digital contracts, and the delivery of e-commerce invoices to end-user, the use of cloud computing has increased significantly, given the wide variety of advantages it provides, such as efficient collaboration, ease of compliance with data security regulations, and simplicity of management and scaling [6]. McAfee discovered that 18.1% of data transferred to cloud-based file-sharing platforms contain sensitive data, such as intellectual property [35]. However, this also gained the attention of attackers, who began targeting cloud infrastructure to launch supply chain attacks. For this reason, CSP encourages the potential customer to utilise threat assessment services and recommend AES-based encryptions for their storage. The widespread adoption of the AES encryption standard is especially concerning because any future assertion to tamper with the AES encryption might have an unenviable impact on the integrity of sensitive information stored by large cloud storage providers.

ChaCha20 cipher is significantly quicker as it leverages Addition-Rotate-Xor (ARX) based instructions throughout each round and is a lot faster than AES software-only implementation for large files. The project objective is to evaluate an alternative encryption algorithm called XChacha20, which is derived from chacha20-based encryption that can be used to perform encryption for various file formats in cloud-based storage. For validating the outcomes, the files in cloud storage are tested results are illustrated, the results of the experiment using various factors, such as the avalanche test (key-sensitivity test), correlation, entropy, time complexity, execution time.

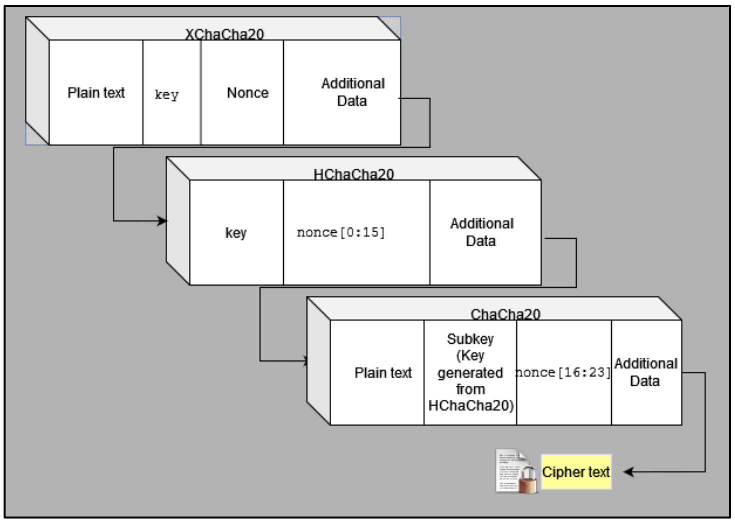


Fig 1: Flowchart of XChaCha20 Design [22].

1.2 Research question

According to previous research on the widely used encryption algorithm known as AES, there were practical nonce misuse attacks in the whitepaper by Hanno [27]. Additionally, because the AES relies on the hardware CPU instruction set. So there is a possibility to perform cache timing attacks if the hardware instruction set for AES is unavailable, this was specified in the research of side-channel attacks conducted by Ashokkumar et al. [5]. To begin with, theoretical proposals for AES based ciphertext attacks are targeted. This allows for a more accurate assessment of how expensive it would be to carry out a successful attack on AES based encryption in future.

Research Question:- How does XChaCha20 perform as an alternative encryption method to AES for file-system-level data encryption in the cloud? As AES is used by all major public cloud providers, the alternative algorithm should provide improved security and do not exert an extra burden on the computing resources of cloud infrastructure.

1.3 Document Structure

The rest of the report structure is as follows:

- **Literature Review:** Section two discusses a brief contrast of existing research in cloud storage security and symmetric cryptographic algorithms. The section also includes a brief report on the reviewed research's challenges, future scope, and limitations.

- **Methodology:** In section three, the research addresses achieving the goal, lists the test data files and the aim to utilise the test data files in this research, and clarifies why this test data file is the best appropriate for this experiment.
- **Design Specification:** This part contains in-depth writing on the architecture, characteristics, functionality, and discussion of the algorithm, among other things.
- **Implementation:** The report will showcase and describe the design of the encryption system, the libraries that were used, and the justification for using the particular libraries and parameters for encryption.
- **Evaluation:** Usage of cryptographic metrics and evaluation are done across AES and Proposed XChaCha20 algorithm and interpret the results to showcase run time execution, entropy, correlation avalanche test and much more.
- **Result and analysis:** Demonstrate results and discuss how the research performed in previous sections answer the research questions.
- **Conclusion and Future work:** Describes how the previous sections of research address the research topics, summarise the total research effort and describe how encryption is actively employed in commercial applications and its limitations.

2 Literature review

This section is divided into two subsections.

Section 2.1 focuses on current research in the cloud and cryptography sections, on how organisations deployed encryption in servers before the emergence of cloud infrastructure, the necessity for encryption in the cloud, and the ongoing evolution in this field.

Section 2.2 aims at weaknesses and shortcomings in current cryptographic techniques. This study will give a course to overcome the existing issues in previous methods, which will then be implemented in the methodology and implementation as part of this research. Including the strategies and problems being resolved also provides an opportunity to discover which cipher algorithm has been used in the past and whether there is a baseline limitation.

2.1 The need of enhanced security in cloud computing infrastructure

Cloud computing is a way of sharing infrastructure through a provider over the Internet. As a result, users get the freedom to simply access their data at any point in time. On the other side, there are concerns about the security of data for various reasons. To begin with, there has been a significant advancement in cloud computing. Additionally, the rising demand for computer capacity has led to a rise in the number of enterprises that outsource their storage space. As a result, cloud storage providers are required to store data as block storage or object, and unencrypted storage is considered as a serious concern like 4TB of data from the Brazilian Health Ministry, which was maintained in a cloud service, was compromised. [40].

As a cloud service, some security measures are essential to achieve the following:

Confidentiality: In [21] researcher states, "*Data confidentiality refers to protection from unintentional, unlawful, or unauthorised access to the data*." It is a solid example of confidentiality since most organisations fall victim to cyberattacks that may lead to sophisticated supply chain attacks to steal confidential files, negatively influencing the organisation's reputation.

Integrity: As stated in a post by Tierney [20], "*A crucial component of cloud data security is data integrity — preventing unauthorised modification or deletion*" One such example is that a change of bytes should not alter or modify the original file. So, it is vital in this test to perform an avalanche test on the encryption model using the test data files to verify the degree of file change. Additionally, Poly1305 generates a 16bit hash to verify the message block before the decryption hash function validates the key and encrypted block.

Authority: "*A sysadmin of the cloud provider that has privileged control over the backend can perpetrate many attacks in order to access the memory of a customer's VM*" As quoted from Towards Trusted Cloud Computing [19], Insider attacks are the most common type of attack, and when it comes to cloud computing, the principle of least privilege applies to a great extent. However, it is not only the consumer who has access to files in the instance. CSP requires access from support agents to system engineers to fix bugs or perform maintenance in the virtual machine.

Block storage, object storage and file storage are three main types of cloud storage that cloud service providers often use. Amazon Web Service (AWS) provides three types of storage: Elastic Block Storage (EBS), Elastic File Storage (EFS) and S3 Buckets. Elastic Block Storage (EBS) is fairly block storage, EFS is a file storage solution it can be mounted to multiple computing instances and is simple to scale as required, while S3 Buckets is object storage and relies on Application Programming Interface (API) to manage files. Storage in cloud is easy to replicate across various zones for many reasons like Data regulation, load balancing, and even as backup solutions. The encryption according to AWS storage standards [39],

- **File-system-level Encryption**: In contrast to filesystem-level encryption, which runs on top of the file system, encrypts the specific folder or files in the disk, they are accessible to adaptable to a wide range of operating systems.
- **Disk encryption**: This is a situation where the whole volume is encrypted and allows the user to mount encrypted volumes to computing instances, service like AWS use Dm-crypt, and DM-Crypt internally performs AES based encryption.

Cloud computing benefits both organisations and consumers as features like CDN and Cross-region replication reduce the load on a single instance. However, it must overcome security flaws to benefit from cloud computing services. Overall, the Research [10] confirms that security is an important problem for both consumers and communication service providers. Because the statement was validated by the literary analysis of multiple research journals, it is recommended that research provide a standard of addressing security rules and standards where data storage is considered a threat in cloud computing.

The author Hyder, Tooba et al. [17] researched a cloud-based encryption approach using OpenStack. The analysis was carried out on an Ubuntu-based machine. New instances are created on the Compute cluster to handle the job, and data is sent over the cloud. The data values in the results are higher than predicted since the analysis [17] is conducted via the Openstack user interface. RSA-SCS module calculated a 16 KB file with three unique security parameters. Using the KeyGen technique takes longer than other methods since the client generates unique two prime numbers first, then another three integers to compute the private and public keys required for encryption and decrypt.

2.2 Shortcoming of existing cryptographic algorithms

The research [7] compared several encryption algorithms, AES, DES, and RSA—based on examining the simulation time for encryption and decryption for each of the three methods. They concluded that the AES algorithm exceeded the DES and RSA algorithms in terms of performance and research tabulation also acknowledged the fact that RSA was concluded to be the least secure of all three algorithms, with AES being the most secure.

Though AES is considered an encryption standard, there were few theoretical and comprehensive attacks like Side-channel attacks, nonce disrespect forgery attacks, and cache-timing attacks on AES encryption. Internally, AES iterates over multiple rounds s-box depending on the key size.

If future cryptanalyst reveals a critical flaw in AES, file protection will become more complex. The only alternative commonly accepted encryption is the slower 3DES therefore, changing encryption deployments to 3DES is not reasonable.

Table 1: AES cipher rounds based on key size

Key size	**Rounds**
128bits (or) 16 bytes	10 rounds
192bits (or) 24 bytes	12 rounds
256bits (or) 32 bytes	14 rounds

The researcher Bogdanov, Khovratovich et al. [11] from Microsoft has also managed to perform full key recovery on AES encryption with all three key sizes using the Biclique Cryptanalysis approach on Meet in the middle attack, a common cryptographic flaw affected algorithms such as triple DES [12]. Table 1 was constructed from the research performed by the author [11].

ChaCha20, based on encryption, is being used as a standard in various protocols and drivers like Transport Layer Security[13], SSH[15] QUIC and Linux kernel, according to the author in [14], this set of fixes increases the entropy collection capabilities of virtual machines operating on Microsoft Azure, and it will take the use of a hardware random number generator (if one is provided) to populate the */dev/random* pool.

Even though ChaCha20 has a limited number of cryptanalysis compared to AES and researchers were able to penetrate up to 7 rounds, the feasibility of brute-forcing key to retrieve the plain text by penetrating more than 20 rounds is infeasible with current computer resource availability. And authors of [37] states that

"Thus, our findings, as in the case of all the state-of-the-art results in this area, do not provide any cryptanalytic threat towards the ciphers with full rounds".

Furthermore, when using XChaCha20, the HChaCha20 algorithm is used to generate subkeys based on the remainder of the key and nonce, a pseudorandom keystream, is generated by Chacha based on the of the subkey and nonce, which is used in the Addition, Rotate, and XOR (ARX) function. This reduces the risk of a none reuse attack, as opposed to AES, which has a limited nonce of 96bit sequence. The overall flow of XChacha20 shown in figure 1 is derived by the whitepaper of XSalsa[18]. The research from the KDDI lab shows the complexity of carrying out successful brute force attacks on ChaCha20 encryption. [29].

The Section 2.3 report will highlight the findings, goals, and limitations from the few interesting research publications that would be necessary to answer the previously proposed research question. All the feedback from the section will be used to consolidate the outcomes of this research and will be used to overcome the constraints researchers have identified in their proposal.

2.3 Summary of Research Studies

Table 2: Literature review summary table

Author	**Title**	**Findings**
Babitha and Babu [7].	Secure cloud storage using AES encryption	This article talks about the need and implementation of encryption in the cloud. The author asserts that encryption in the cloud can provide **confidentiality, integrity, availability, and ease of data sharing**.
Koo et al. [31]	Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage	With the following Research, one can get a solid need for **attribute-based encryption** a decent degree of accuracy on the **encryption architecture**. Trusted Authority, Cloud Service Provider (Amazon and Azure), Data Owner, and Retriever are terms used in this research context.
Daniel J. Bernstein. [34]	ChaCha, a variant of Salsa20	Daniel is the author who designed Salsa and Chacha encryption from scratch. The research paper gives a great amount of **transparency on algorithm design**, calculation of **quarter rounds** and **transformation of the key** under ARX. This paper is also an ultimate comparison between his previous algorithm called Salsa.
Aumasson, Fischer et al. [25]	New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba	The journal paper shows a **cryptanalysis investigation of the Chacha, Salsa, and Rumba**. On all three methods, the study performs Differential Analysis, computes the backward probabilities, and briefly describes the time complexity [25].

Thabit, Alhomdy et al. [24]	Security analysis and performance evaluation of a new lightweight cryptographic algorithm for cloud computing	This research compares across various algorithms and, importantly, the metrics of the algorithm, this research offers an immense level of **knowledge on cryptographic metrics** and their **overall performance** for the cloud. The data files and tests are limited to image file format
Mahmoud, Darwich et al. [33]	Cost-Efficient Storage for On-Demand Video Streaming on Cloud	The comparison in this research focuses on the cost of cloud computing for storing and transcoding video content. The study's findings assisted in better understanding the **transcoding system offered by cloud providers** as well **as media management in cloud-based storage** environments.
Böck, Zauner et al. [27]	Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS	Nonce abbreviates for **number** only used **once**, and this research shows a theoretical demonstration on the impact of reusing the same **nonce in TLS communication using AES symmetrical encryption**. The attack has **practically been demonstrated in Defcon** 2016 (https://www.youtube.com/watch?v=uxuXFK5XKEU) by the authors, So proposed encryption needs to be resistant towards nonce misuse.
Crowley and Biggers [36]	Adiantum: length-preserving encryption for entry-level processors	Researcher Paul and Eric from Google suggested the HBSH algorithm for Adiantum, which employs the XChaCha12 variant with Poly1305, the benchmark compares it to the AES algorithm. According to a benchmark of earlier findings, the study outlines the restriction of utilising a maximum of 2^{55} bytes of plain text for each encryption cycle with a single key, and the time needed for creating ciphertext of such size is 11 years, as shown in the tabulation.

2.4 Research Gap

After going through the literature, it is clear there is a must of encryption in cloud storage, and all the major cloud providers rely only on AES for encrypting the users' file system. These files format can be multimedia, text, archive. Considering the possible attacks on AES theoretically, as stated by researchers [5] [11], it is essential to propose an alternative algorithm and compare an encryption system for a cloud that eliminates the risk of nonce reuse, with easier to adapt and secure from modern-day cryptanalysis.

XChaCha20 with Poly1305 AEAD will be alternative encryption to AES, XChaCha20 takes 256-bit keys, and 192-bit nonce, which internally generates a unique subkey uses a part of the nonce for the internal ChaCha20 algorithm. This makes the whole algorithm resistant to nonce misuse. As per the specification of XChaCha20 Poly1305 of Internet Engineering Task Force – IEFT [38]

"*Assuming a secure random number generator, random 192-bit nonces should experience a single collision (with probability 50%) after roughly 2^96 messages (approximately 7.2998163e+28). A more conservative threshold (2^-32 chance of collision) still allows for 2^80 messages to be sent under a single key.*"

This is much larger than Paul Crowley's HBSH limitation on maximum text size for the single key with the XChaCha12 variant of the algorithm.

3 Research Methodology

This part of the document focuses on the methodology for achieving the research goal and is classified into three different sections.

3.1 Test Data Files

Cloud storage can handle a variety of file types. In order to be considered the test reliable, the test findings must be validated in a broader context, support a variety of file formats, and can encrypt files of varying sizes. Consequently, the files selected are in four distinct file formats with file sizes ranging from kilobytes to gigabytes.

Table 3: Test Data Files for evaluation

Name	File format	Size
Baboon	png (image/png)	623K
Big Bunny bucks	mp4 (video/mp4)	31M
Linux Kernel	zip(application/zip)	230M
Ubuntu 20.3	iso(application/octet-stream)	2.9G

Because the research topic concerns cloud storage security, it is critical to run the tests on cloud providers to understand the results better. The samples stated above will be subjected to a series of encryptions and decryptions using the XChaCha20 and AES algorithms, respectively.

3.2 Evaluation Metrics

Avalanche Test: Avalanche is a metric used to calculate the randomness and sensitivity of the algorithm. This is commonly calculated by flipping in either key or plain text [24]. In our case, changing a bit in plain text will affect the file's integrity, so a bit from the key will be flipped or incremented. A batch of encryption is performed after incrementing the bit. Another batch of encryption is performed to calculate the avalanche effect.

Entropy: Entropy is a metric of the information for data quality, which implies how difficult it is to retain the original data after high random [42]. Figure 5 represents the entropy generated for a ZIP file, and it can be seen that a few bytes of data are allocated randomly over an unencrypted file and that the majority of the underlying difference is hidden. The overall image is normalised for an encrypted file, the entropy values obtained from these two methods are very similar to one another, as can be seen by the entropy value mentioned.

Execution Time: Time is the amount of time necessary for an operation to finish in its environment. It depends on several factors, such as the algorithm, the IO bandwidth of the drive, and the speed of computing resources.

Time Complexity: It takes a long time to generate a variety of keys and use those keys to carry out an attack, which is a cryptographic characteristic known as time complexity. For example, let us assume that the time required to test a key for decryption is O(1) in the Big O notation.

Keyspace complexity: Key-space is the number of distinct keys required to perform a brute force attack. In general, these keys are generated in a sequence of order. For example, In order to successfully brute-force a 20-bit key length, the attacker will need to generate million possible keys as shown in figure 2.

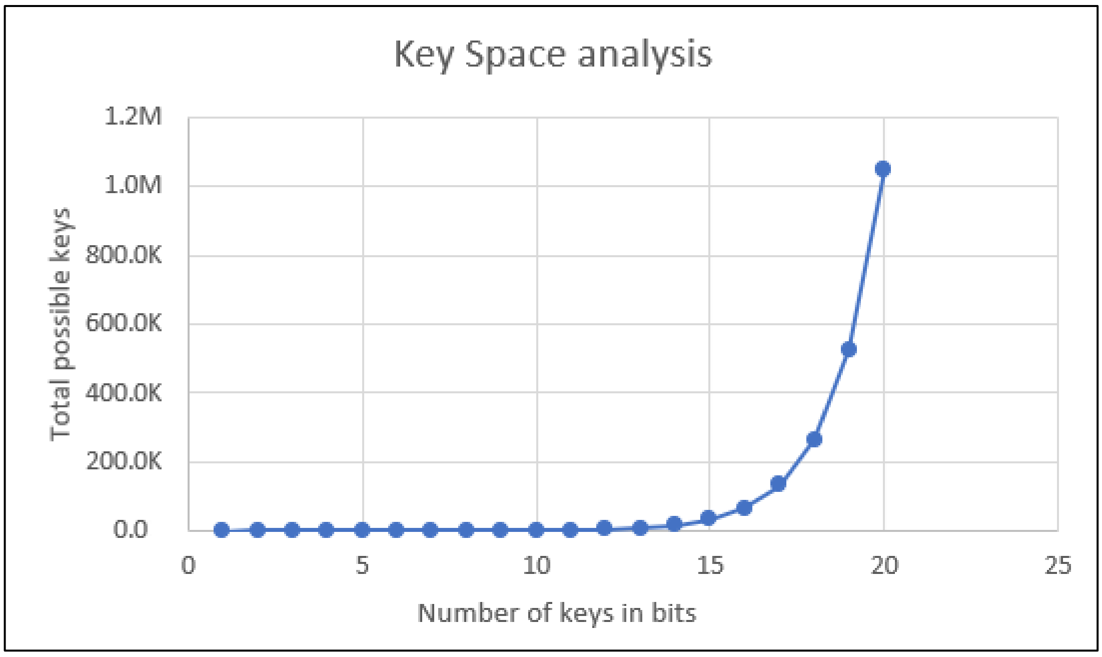


Fig 2. Keyspace analysis for 20bits

3.3 Cloud File Security Parameters

When it comes to cloud implementation, it is essential to examine a few criteria such as resource consumption during encryption and decryption, the size of the file after encryption, and the security of storing and retrieving the encryption key and randomness to generate a nonce.

Trusted Platform Module (TPM) is a hardware chip responsible for securely handling encryption, Cloud providers using AMD based EPYC and Intel-based XEON CPU's have support for TPM Module. Many applications like **BitLocker** for windows, **Dm-Crypt** for Linux, and Digital Rights Management (DRM) tools rely on TPM [43] for secure encryption and decryption. Because TPM needs to implement the ChaCha based algorithm in order to handle XChaCha20 cryptography at the volume level, it is required to keep the key for the volumes inside the TPM module, a procedure known as binding a key that secures the key from disclosure [41]. The current version of TPM 1.2 and V2 supports AES128 and AES 256. The software-only version of XChaCha20 is used to perform evaluation, perform file-system level encryption, and do not extensively rely on specific hardware instructions like AES NI.

It is critical for the management of keys and nonce that the key created is maintained securely by the cloud provider. Thus, the service provider needs to use an isolated module like TPM for volume level encryption. Therefore, if the CSP vendor wants to setup an encryption file system level application, they can rely on storing the key to a database.

3.4 Working of XChaCha20

The functionality of XChaCha20 is derived from the XSalsa20 algorithm [18], XChaCha20 consumes the following algorithm:

- A set of the unique 256-bit key is generated and is considered as K1
- L1 is a nonce with a length of 192 bits and is randomly generated. The first 128 bits of L1 are utilised to generate the HChaCha subkey.
- HChaCha20 generates a 256-bit subkey K2 from the input keys K1 and L1[0:15].
  K2 = H (K1, L1[0:15])
- Padding the left out nonce is necessary. Let us consider it as L2.
  L2 = padded_nonce(L1[16:23])
- Using ARX, each quarter cycle of the ChaCha algorithm generates a block cypher named keystream key stream= ChaCha(K2, L2)
- Each byte of Plain text is XOR'ed with keystream, ChaCha-based algorithm is considered a stream cypher. (Ciphertext = keystream $\oplus$ Plain text)

4 Design and Specification

4.1 Purpose

This design and specification will detail the procedures and pre-requisites for implementing the XChaCha20 over the file system layer in cloud instance and the architecture of the encryption algorithm.

4.2 System overview

Since the files to encrypt are in a cloud instance, it is very significant to containerise keys for encryption in isolated architecture, so if the application relies on the File system layer, user's can not directly interact with TPM module to perform encryption and applications can not access the storage root key or master wrapping key to decrypt, In such scenario best use-case will be utilising software only Key Management Server (KMS), that extensively uses the isolated database to store the keys for the files in File System.

4.3 Levels of encryption in storage level

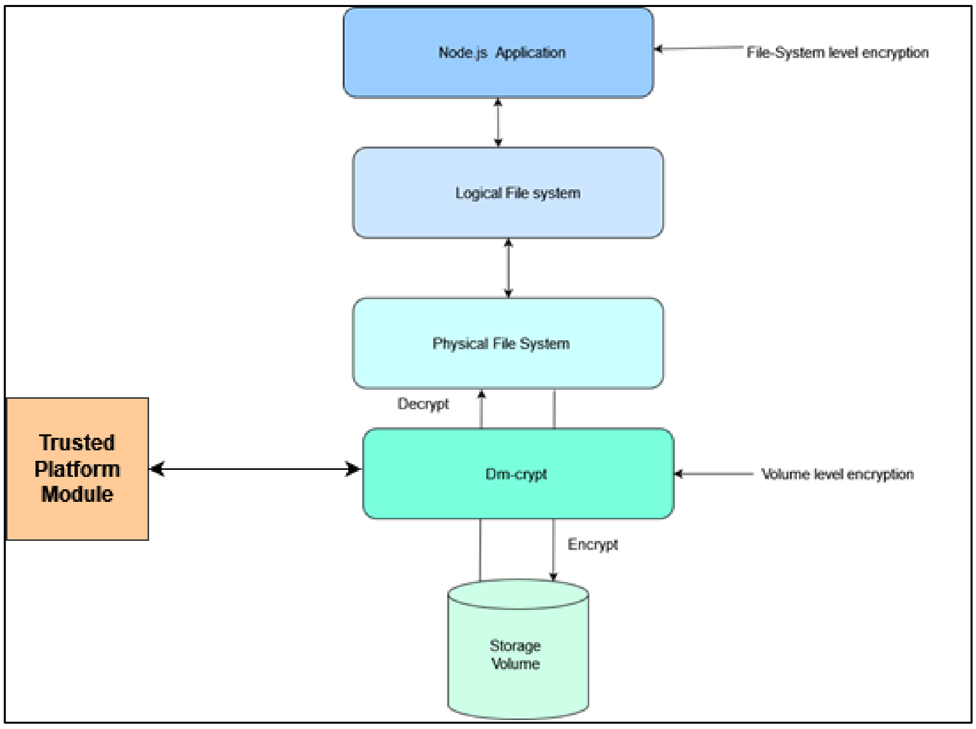


Fig 3. Encryption level in Disk system

The reason to perform file system level encryption as illustrated in figure 3 is because encryption at application level or file system level cuts off security implication at the bottom levels of the hardware and kernel stack, So if computing instances is affected due to vulnerabilities like RCE or OS Command Inject attacker will be able to access the data of the encrypted volume as files in the disk will be in a mount state and Dm-Crypt would not be sufficient for avoiding the breach.

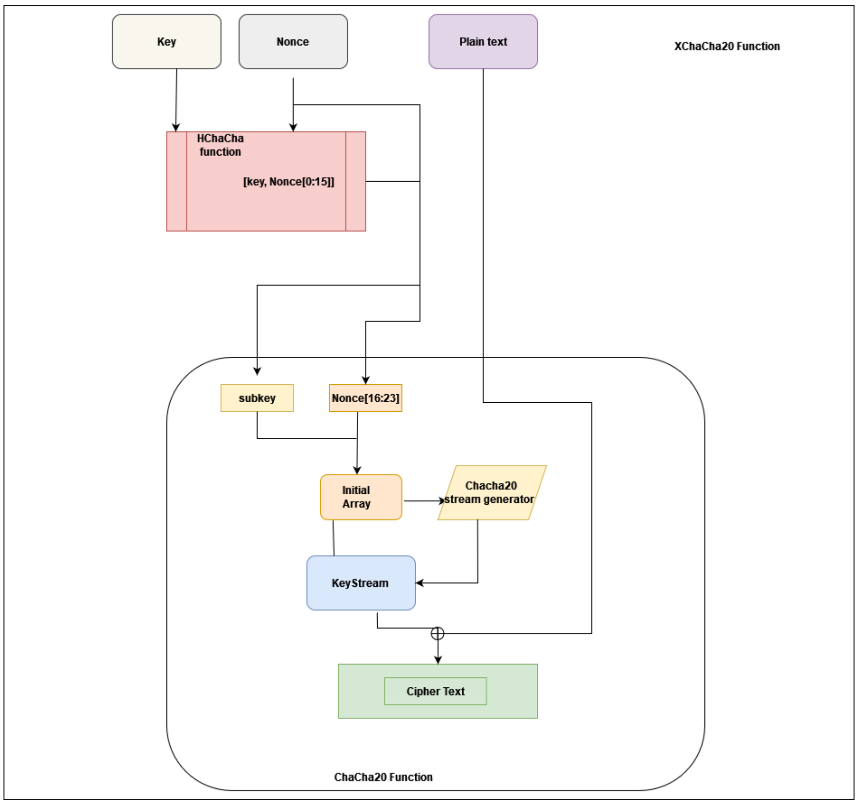


Fig 4. XChaCha20 Functionality [22]

The architectural design of XChaCha20 is shown in figure 4, and the flowchart is adapted from the XChaCha20 libsodium implementation [22].

5 Implementation

The work was carried out with the help of the Libsodium implementation of XChaCha20 [22] and the Software Only Implementation of AES256 to analyse the XChaCha20 performance and metrics of the test data file specified in section 3.1. The decision to use a software-only version of AES was made to do a fair computation on the algorithm's speed.

5.1 NPM and NodeJS

Javascript is a general paradigm programming language. The encryption suite is developed in JavaScript because, according to the Stack overflow 2020 survey, "JavaScript completes its ninth year in a row as the most commonly used programming language" [23]. As a result, it is essential to validate the tests in the most widely used programming language.

NodeJS is a powerful open-source JavaScript run environment used to build robust applications. NodeJS introduces an event-driven paradigm to implement the non-blocking asynchronous operation, allowing developers to write asynchronous code in JavaScript. Using clustered programming and libUV, NodeJS developers can build modular applications that utilise multi-cores of the instance for concurrent processing.

In addition to the convenience of using a scripting language (JavaScript), NodeJS provides library support for networking, file system IO, buffers, webservers, and cryptographic programming and many other features.

NPM is the package manager used here to manage the libraries and dependencies for NodeJS. A libsodium wrapper [22] for Javascript is used for validating XChaCha20 algorithmic implementation.

Upon encryption with XChaCha20, the file extension is changed to *.enc*, the key, nonce and FileName name is stored as JSON format to the *prem.key* file, which is later deserialised during decryption. The nonce and key are deserialised to pass as parameters to the decryption function, including the path of the encrypted file. In the case of AES encryption, the encryption module saves the file with *.aes* extension and the key file is used to store the required parameters for decryption. Given that NodeJS is capable of running on both x86 and x64 bit based devices, and that it supports essentially all modern operating systems, the implementation of encryption and decryption may be handled with relative ease on a broad variety of platforms.

5.2 Cloud specifications

The full implementation and analysis were conducted on three distinct platforms, Amazon Web Services (AWS), Microsoft Azure, and a Virtual box. Each of the three virtual computers was created with the same configuration.

Table 4: Implementation of Virtual machine platform and specification

Platform	OS	Kernel	CPU	Memory	Category
AWS	Ubuntu 18.04.6 (Minimal)	5.4.0-AWS	Intel Xeon E5-2686 v4 (2) @2.299GHz	8GB	M5.large

Azure	Ubtuntu 18.04.6 (Minimal)	5.4.0-Azure	Intel Xeon E5-2673 v4 (2) @2.294GHz	8GB	D2.v4
Virtual Box	Ubtuntu 18.04.6 (Minimal)	4.15.0 generic	Intel i7-4710HQ (2) @2.49Ghz	8GB	

6 Evaluation

To validate the results and compare against the standardised algorithm for encryption – AES. This will help us to evaluate the efficiency and security of the algorithms, and the XChaCha20 will be evaluated against the following metrics.

6.1 Avalanche test

ASCII representation: "177"   Binary representation : 10110001 // increment by 1bit
Altered ASCII : "178"   Binary representation : 10110010
Position altered : 25th

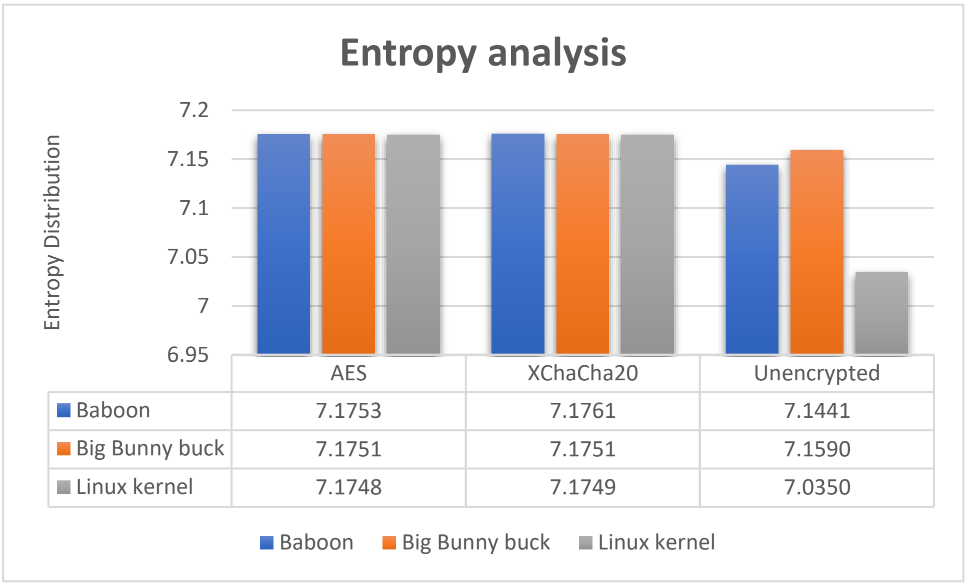
Table 5: Avalanche Effect table

Original_key	Altered_key	File	Plain Bytes	XChaCha original	XChaCha altered	Percentage Of Change	Avalanche analysis	AES original	AES altered	Percentage Of Change	Avalanche analysis
… 32 31 20 31 37 **37…**	… 32 31 20 31 37 **38…**	Baboon	89 50 4e 47 0d 0a …	0d fc 3e 4d 9a 78 …	db 3a c6 3f fb 4a …	99.62508	0.003754	9e 04 62 18 12 6a …	ce b9 0b 07 36 2a …	99.61644	0.003835
… 32 31 20 31 37 **37…**	… 32 31 20 31 37 **38…**	Linux kernel	50 4B 03 04 0A 00 …	70 df 5a af 6b ba …	02 21 8b 7c fc 40 …	99.60940	0.003905	48 1f 2f 5b 15 60 ….	18 a2 46 44 31 20 …	99.60946	0.003905
… 32 31 20 31 37 **37…**	… 32 31 20 31 37 **38…**	Big Bunny buck	00 00 00 20 66 74 …	84 AC 70 2A F1 06 …	52 6A 88 58 90 34 …	99.60918	0.003905	47 E9 45 60 5D 54 …	17 54 2C 7F 79 14 …	99.60839	0.003916
… 32 31 20 31 37 **37…**	… 32 31 20 31 37 **38…**	Ubuntu 20	45 52 08 00 00 00 …	C1 FE 78 0A 97 72 …	17 38 80 78 F6 40 …	99.60956	0.003904	02 BB 4D 40 3B 20 …	52 06 24 5F 1F 60 …	99.60929	0.003907

From table 5, It is observed that XChaCha20 has a great degree of change on altering the key, and the outcome is the same for AES. To investigate further, Have to calculate the avalanche analysis . Avalanche analysis is calculated by (Number of Changed bit in ciphertext) /(Number of bits in ciphertext).

6.2 Entropy

Table 6: Tabulation of observed entropy value



	AES	XChaCha20	Unencrypted
🟦 Baboon	7.1753	7.1761	7.1441
🟧 Big Bunny buck	7.1751	7.1751	7.1590
⬜ Linux kernel	7.1748	7.1749	7.0350

From figure 5, That data of Linux Zip archive in the unencrypted stage has a huge variation. The reason is that bytes in the file is not uniformly distributed. Upon encryption, The file is normalised with a very negligible variation of the difference between each other. Entropy level for XChaCha20 column is higher than AES and unencrypted, as shown in Table 6.

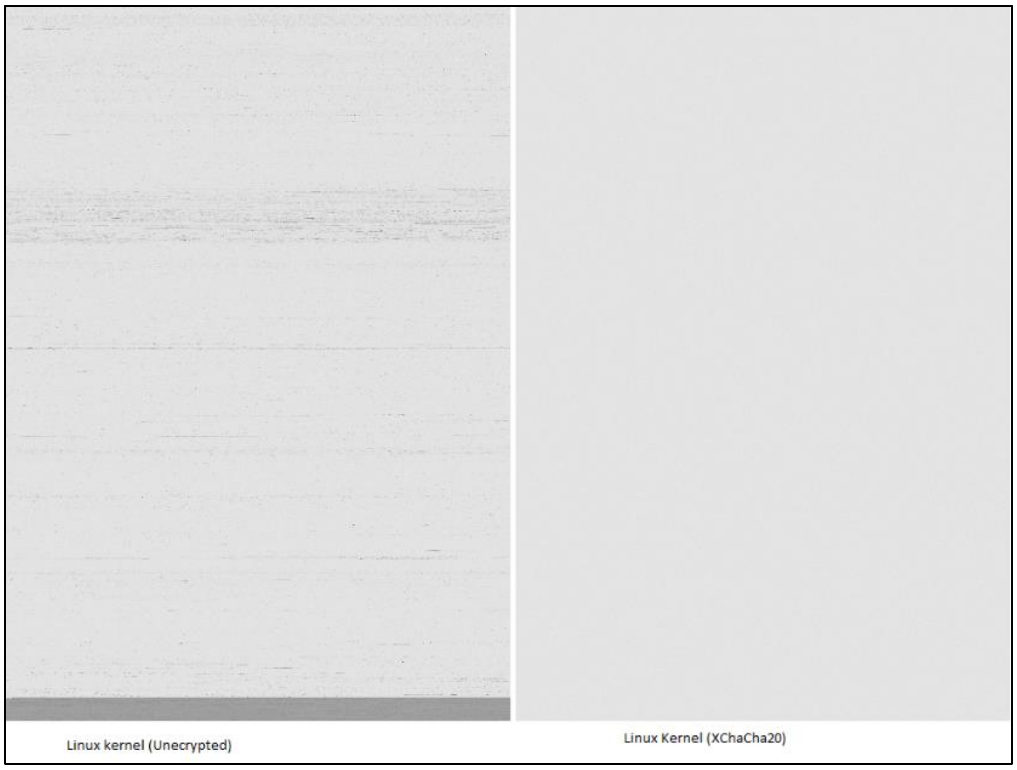


Fig 5. Entropy distribution of Linux kernel test data files

6.3 Execution time

From grouped data in table 9 on analysing throughput of the results, This shows the average throughput and processing chunks in XChaCha20 is comparatively better than AES256 (Software only implementation).
The throughput was measured in Megabytes/second, which was derived by dividing the entire file size in megabytes by the total number of seconds it took for each individual algorithm to complete.

Table 7: IO Throughput across different encryption.

			Throughput (MB/s)		
File	**Mode**	**Algorithm**	**AWS**	**Azure**	**Virtual Box**
baboon	Encrypt	AES	3.53	3.65	3.48
		XChacha20	4.37	4.05	3.66
	Decrypt	AES	3.56	3.64	3.38
		XChacha20	4.10	4.21	4.05
big_bunny_bucks	Encrypt	AES	24.41	23.37	27.21
		XChacha20	39.13	38.70	37.02
	Decrypt	AES	24.44	23.05	26.52
		XChacha20	37.90	37.33	38.37
Linux-kernel	Encrypt	AES	26.94	26.47	28.88
		XChacha20	41.71	42.57	45.22
	Decrypt	AES	27.13	25.31	29.72
		XChacha20	44.23	43.77	46.90
ubuntu-20.04.3	Encrypt	AES	20.19	23.20	20.71
		XChacha20	30.36	37.50	28.84
	Decrypt	AES	28.04	25.74	31.07
		XChacha20	26.87	35.86	30.69

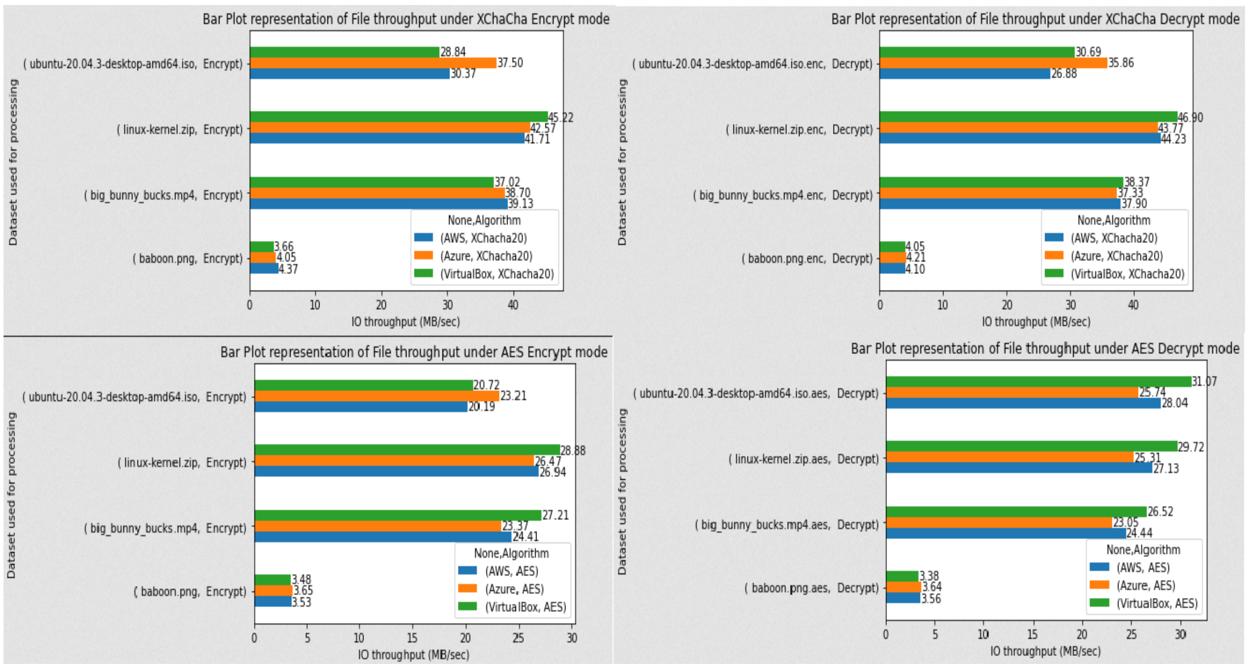


Fig 6: Bar Plot comparison with cipher-algorithm with mode and computing instance provider.

Additional trials were conducted based on file type and environment to determine how the overall throughput was affected by the CPU's clock speed, and the result is bar plotted in figure 6. As it can be seen in the results, a few tests were unexpected, such as when Ubuntu ISO was encrypted in Azure instance with a higher throughput when compared to PC and the Amazon cloud. All three instances were equipped SSD based storage.

6.4 Time Complexity

According to the author from [25], who performed analysis to find optimal differentials on a reduced seven rounds of ChaCha, the time complexity was estimated to be O(2^{248}) on a 256-bit key. As XChaCha20 uses ChaCha20 internally, the estimated time complexity for XChaCha20 is similar.

6.5 Keyspace analysis

XChaCha20 has a 256-bits key size, so the attacker needs to generate 1.158 x 10^{77} (2^{256}) keys to perform successful brute force, which assumes the attacker has the right nonce to perform decryption. Considering the required keys for the attack, brute force is the infeasible and inefficient way to crack modern cyphers like XChaCha20.

6.6 Correlation

Table 8: Correlation table of three Test Data Files

(Baboon)	XChaCha20	AES	Unencrypted
XChaCha20	1	-2.70E-05	-2.51E-03
AES	-2.70E-05	1	-7.24E-04
Unencrypted	-2.51E-03	-7.24E-04	1

(Big Bunny buck)	XChacha20	AES	Unencrypted
XChaCha20	1	-2.80E-04	-3.30E-04
AES	-2.80E-04	1	-9.20E-05
Unencrypted	-3.30E-04	-9.20E-05	1
(Linux Kernel)	XChaCha20	AES	Unencrypted
XChaCha20	1	6.00E-06	9.40E-05
AES	6.00E-06	1	-4.30E-05
Unencrypted	9.40E-05	-4.30E-05	1

The lower the correlation shows they are less linear to each other. This can also be scatter plotted to find a linear relationship between two sets of files. In figure 7, hexbin plot is used to plot the variables to understand. While test data files have a large number of data points, a hexbin plot may be used to depict the connection between two numerical variables where continuously overlapping points are highlighted with green points.

Thus, one can observe AES for both the plotted graph big bunny and correlation plot has frequent overlapping bytes across Unencrypted file. Whereas XChaCha20 is highly distributed in hexbin plot.

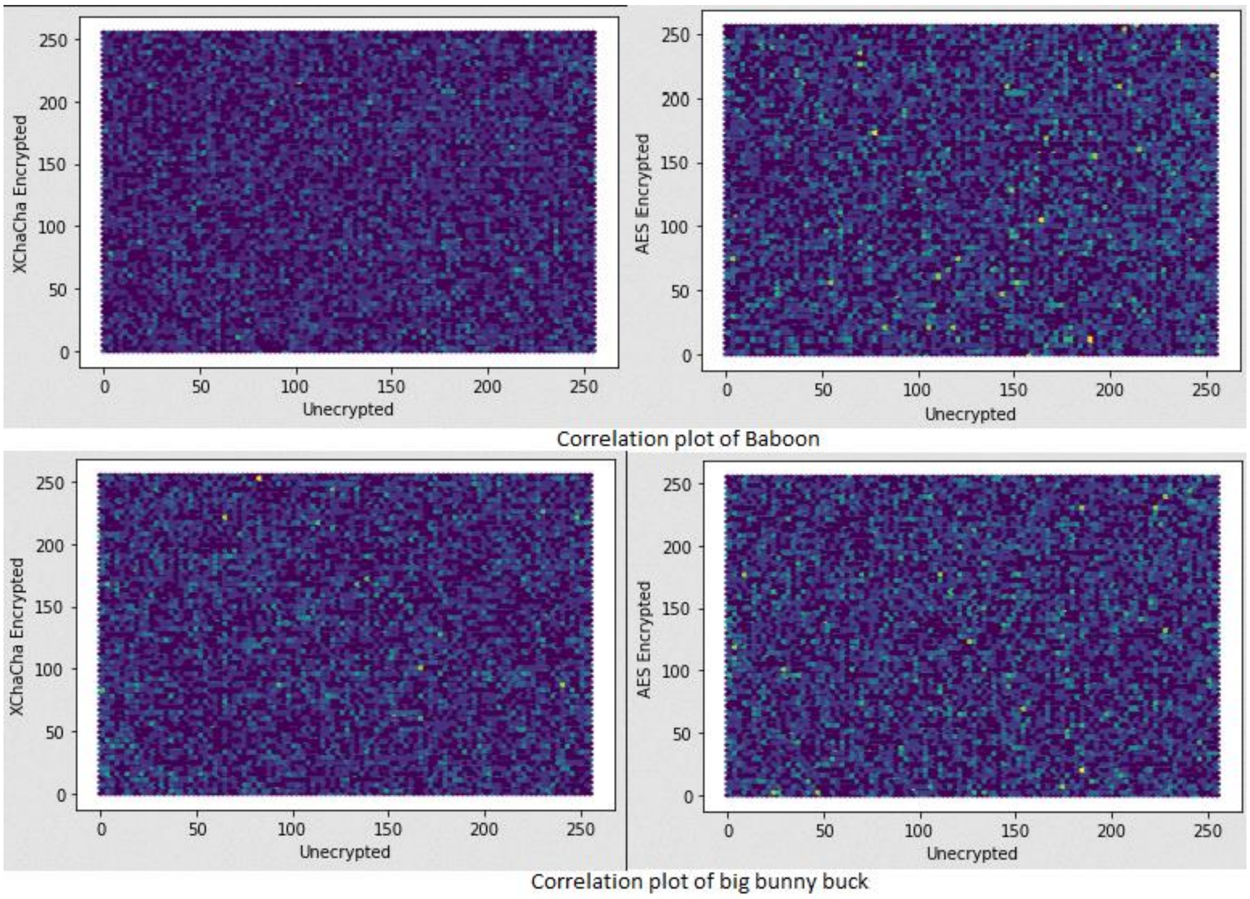


Fig 7. Correlation Analysis plot between encrypted and unencrypted parameters.

So the major takeaway from figure 7 correlation analysis is to understand there is a higher correlation of bytes with dense plots at specific regions in AES and unencrypted, in contrast to XChaCha20 vs unencrypted, this dense bytes increases the risk of frequency analysis attack.

6.7 Runtime analysis:

6.7.1 Overall Run time

The CPU and memory monitoring test was performed where the Ubuntu20 file was processed under supervised resource consumption, and it is seen the overall encryption and memory are not resource-hogging for the computing instance. The overall time consumption was 125000ms because the memory and CPU management script was processing the encryption under child process. In figure 8, The Y-axis is the percentage of resources used, X is a series of total times while encryption took place, the program was streaming the bytes to a buffer chunk of 600MB. Similarly, to the personal computer, resources for the below test have been assigned using a profiler. Profiler used: Syrupy (System Resource Usage Profiler).

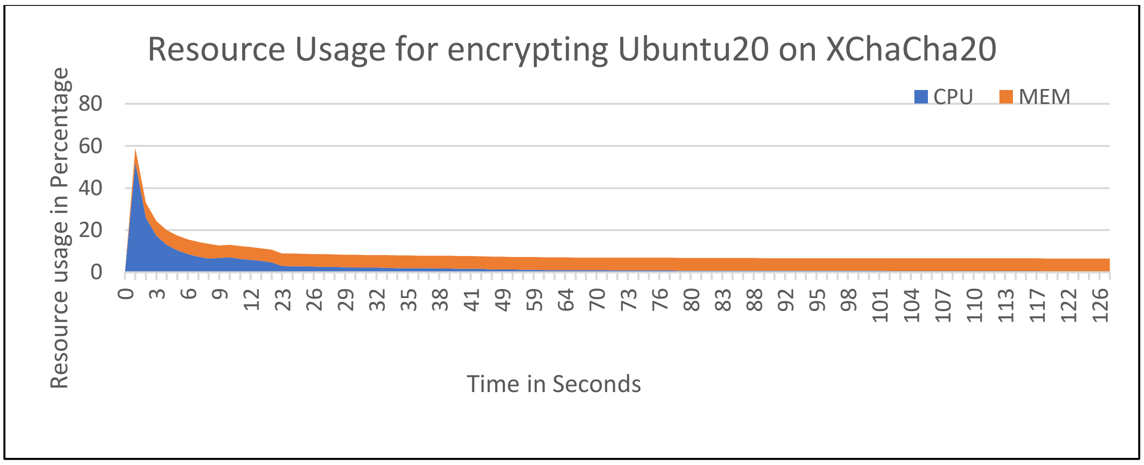


Fig 8. Resource usage of 2 CPU and 8 GB RAM in VirtualBox.

Table 11 presents an average summary of the five rigorous tests of encryption and decryption on different cloud vendors. Figure 6 shows graphical comparisons between the different cloud platforms and the total duration of the encryption followed by decryption over five times.

Table 9 : Standard Deviate and Run Time average

Encryption	**Mode**	**Statistics**	**AWS**	**Azure**	**Virtual Box**
XChaCha20	Encrypt	Mean	28.90	30.71	28.69
		Standard Deviation	17.05	17.90	17.97
	Decrypt	Mean	28.28	30.29	30.00
		Standard Deviation	17.64	17.72	18.53
AES256	Encrypt	Mean	18.77	19.17	20.07
		Standard Deviation	10.53	10.46	11.61
	Decrypt	Mean	20.79	19.44	22.67

		Standard Deviation	11.59	10.60	13.00

Also, we can see that the performance of some tests was noticeably better in the personal computer because the CPU in the cloud instance had a lower clock speed than the PC and that the Azure encryption with XChaCha20 throughput was noticeably faster than the AES encryption, which was particularly noticeable for large files encryption.

Five concurrent tests were conducted on all test data files to generate an average result. The overall goal of this test was to establish an average benchmark or IO throughput that an encryption system is capable of with the same set of instance configurations. This can be seen in the following chart: files with the extension "*.enc" are encrypted by XChaCha20 Poly 1305 digest. However, files with the extension ".aes" that used AES 256 bit encryption processed slower than files with the extension ".enc".

6.7.2 Impact of Runtime with controlled resource

To examine further on runtimes, The Research calculated the run times by altering the CPU cores and assigning RAM to VirtualBox. The observation from this study reveals while encryption and decryption are not resource hogging as described in figure 6 but the additional RAM and CPU is used to process quicker if available, So the approach does not bottleneck to certain volume of Memory or CPU.

Table 10 : Tabulation of Average time taken for Ubuntu file with respect to CPU and RAM

CPU	RAM	Time (MM:SS)	Time in sec
4	8	01:32	92
4	6	01:34	94
4	4	02:20	140
4	2	02:18	138
3	8	01:31	91
3	6	02:01	121
3	4	02:20	140
3	2	02:42	162
2	2	02:44	164
2	4	02:43	163
2	6	01:48	108
2	8	01:50	110

Following on from the previous tabulation 12, it was noticed that when the machine's RAM was 8 Gigabytes, the algorithm runtime was less than 120 seconds. In figure 9, when the scatter plot of run time in seconds and RAM allocated for the instance was plotted, it was discovered that linearity existed between the two data points.

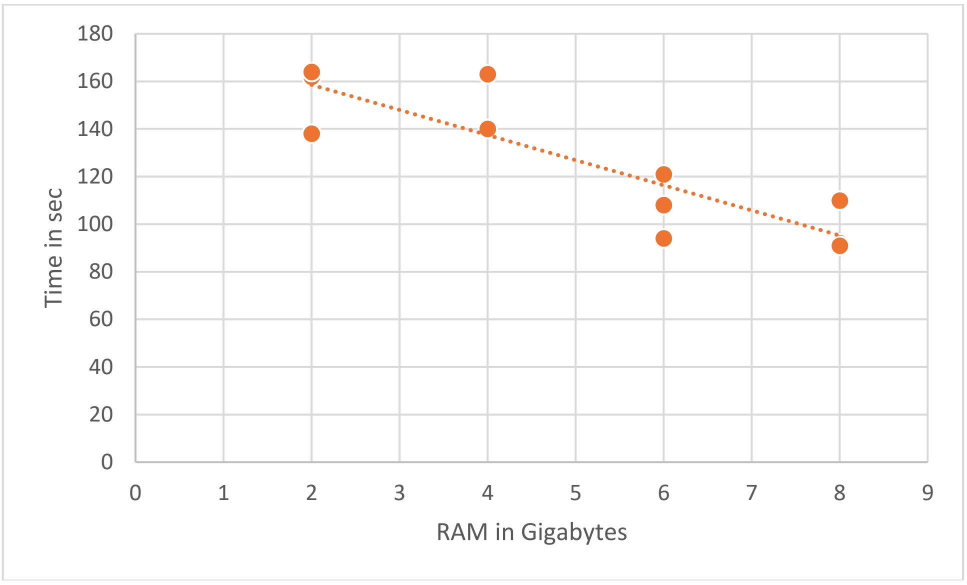


Fig 9: Scatter plot between run time and instance memory

Based on the above metrics like Entropy, Avalanche, Correlation and Runtime. Its evident to see that XChaCha20 can perform better in aspects of randomness, speed, and stability than AES.

7 Discussion

This research presents an evaluation for XChaCha20 that has been actively employed in many commercialised applications [28][29][30] with increased use of nonce than ChaCha20, XChaCha20 leverages the security of replay attack, it also supports AEAD structure.

The above metrics in evaluation section like keyspace analysis, time complexity [25], correlation, entropy, execution time and avalanche tests shows XChaCha20 can provide sustainability as much as or better in some case than AES256 has to offer for file encryption at the same time also addresses nonce misuse attack and does not extensively rely on specific hardware instruction set like AES NI.

The research carried out covers the following section, which extensively answers the research question

- How XChaCha20 is an excellent alternative to AES in terms of evaluation metrics like execution time, correlation, keyspace analysis, Avalanche test and entropy.
- Explained current AES attacks and why an alternate algorithm based on ChaCha20 should be considered because of overreliance on a single algorithm.
- Hardware-independent and software-only solution, allowing the algorithm availability across numerous runtimes, including NodeJS.
- Reported Runtime analysis of file systems in various cloud environments such as AWS (and) Azure and virtual machines with limited hardware resources.

So, In reference to the research question, research has shown how XChaCha20 performs as an alternative encryption algorithm to AES in general cryptography metrics. In terms of improved security that necessitates cryptanalysis, research has cited actual authors like Daniel J Bernstein and other previous work on ChaCha20 Algorithm, which describes how chacha20 is resistant towards Differential Analysis, Linear Cryptanalysis, and side-channel attacks like Cache Timing Attacks. In addition, the literature review section cites practical side-channel attacks against AES that have been carried out in the past. Furthermore, the section on runtime analysis of the report justifies the resource utilisation in the computing servers during the XChaCha20 Algorithm. So the research question is answered to a huge extent.

Limitations: The advancement of encryption algorithms also benefits ransomware developers for example, the most current version of Maze ransomware [38] used a ChaCha-based algorithm to encrypt user's drive files in system level. In general, the advancement of cipher algorithm increases the difficulties associated with key extraction from memory, file decryption and analysis as a constrain there is no possible way to limit the encryption for non-malicious intentions.

8 Conclusion and future work

While our study demonstrates that there are several reasons to consider XChaCha20 as a viable alternative to AES, the majority of cryptanalysis undertaken so far has relied on ChaCha20, which is utilised internally and is likewise resistant to such cryptanalytic assaults [29]. As a result, there is a future opportunity to do cryptanalysis tests on the XChaCha20 Poly1305 cipher, such as the Meet in the middle attack, Adaptive chosen Plaintext Analysis, Cipher only analysis, and chosen plain text analysis.

The encryption scope may be expanded and could be used to encrypt the whole cloud storage volumes or implement encryption module based on XChaCha20 as fallback encryption mechanism.

So remarking all the aspects into consideration and to answer the research question, After all of the evaluations, it is evident that XChaCha20 performed significantly better than AES256 in several metrics. As a result, CSP's can consider using XChaCha20 for encrypting data in file-system level encryption and Poly1305 for generating the authentication digest of the file block as an alternative to Poly1305. In addition, as seen in figure 8, basis of the analysis, the computational burden on infrastructure is relatively minimal.

References

[1] "Encryption at rest in Google Cloud | Documentation", *Google Cloud*, 2021. [Online]. Available: https://cloud.google.com/security/encryption/default-encryption. [Accessed: 8- Dec- 2021].

[2] "The importance of encryption and how AWS can help | AWS Security Blog", *Amazon Web Services*, 2021. [Online]. Available: https://aws.amazon.com/blogs/security/importance-of-encryption-and-how-aws-can-help/. [Accessed: 8- Dec- 2021].

[3] "Azure encryption overview | Microsoft Docs.", *Docs.microsoft.com*, 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/security/fundamentals/encryption-overview. [Accessed: 8- Dec- 2021].

[4] A. Adomnicai and T. Peyrin, "Fixslicing AES-like Ciphers", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 402-425, 2020. Available: 10.46586/tches.v2021.i1.402-425.

[5] C. Ashokkumar, R. Giri and B. Menezes, "Highly Efficient Algorithms for AES Key Retrieval in Cache Access Attacks", *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016. Available: 10.1109/eurosp.2016.29.

[6] M. Gokarna, "Reasons behind growing adoption of Cloud after Covid-19 Pandemic and Challenges ahead", *arXiv*, 2021.

[7] M.P Babitha and K. Babu, "Secure cloud storage using AES encryption", *2016 International Conference on Automatic Control and Dynamic Optimisation Techniques (ICACDOT)*, 2016. Available: 10.1109/icacdot.2016.7877709.

[8] "Encryption On AWS - Tutorial", *Encryption-ws.workshop.aws*, 2021. [Online]. Available: https://encryption-ws.workshop.aws/s3/serverside/sses3.html. [Accessed: 9- Dec- 2021].

[9] N. vurukonda and B. Rao, "A Study on Data Storage Security Issues in Cloud Computing", *Procedia Computer Science*, vol. 92, pp. 128-135, 2016. Available: 10.1016/j.procs.2016.07.335.

[10] B. Alouffi, M. Hasnain, A. Alharbi, W. Alosaimi, H. Alyami and M. Ayaz, "A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies," in *IEEE Access*, vol. 9, pp. 57792-57807, 2021, doi: 10.1109/ACCESS.2021.3073203.

[11] A. Bogdanov, D. Khovratovich and C. Rechberger, "Biclique Cryptanalysis of the Full AES", *Lecture Notes in Computer Science*, pp. 344-371, 2011. Available: 10.1007/978-3-642-25385-0_19.

[12] R. Pich, S. Chivapreecha and J. Prabnasak, "A single, triple chaotic cryptography using chaos in digital filter and its own comparison to DES and triple DES", *2018 International Workshop on Advanced Image Technology (IWAIT)*, 2018. Available: 10.1109/iwait.2018.8369682.

[13] W. Chang, "ChaCha20-Poly1305 Cipher Suites for Transport Layer Security", *Datatracker.ietf.org*, 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7905.

[14] T. Ts'o, "LKML: [GIT PULL] /dev/random driver changes for 4.8", *Lkml.org*, 2021. [Online]. Available: https://lkml.org/lkml/2016/7/25/43.

[15] M. Albrecht, J. Degabriele, T. Hansen and K. Paterson, "A Surfeit of SSH Cipher Suites", *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. Available: 10.1145/2976749.2978364.

[16] Y. Nir, "ChaCha20 and Poly1305 for IETF Protocols", *Datatracker.ietf.org*, 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7539. [Accessed: 13- Dec- 2021].

[17] M. Hyder, S. Tooba and Waseemullah, "Performance Evaluation of RSA-based Secure Cloud Storage Protocol using OpenStack", Engineering, Technology & Applied Science Research, vol. 11, no. 4, pp. 7321-7325, 2021. Available: 10.48084/etasr.4220

[18] D. Bernstein, "Extending the Salsa20 nonce", *cr.yp.to*, 2011.

[19] N. Santos, K. Gummadi and R. Rodrigues, "Towards Trusted Cloud Computing", 2010.

[20] M. Tierney, "Data Security in Cloud Computing: Key Components", Blog.netwrix.com, 2021. [Online]. Available: https://blog.netwrix.com/2020/07/02/cloud-data-security/.

[21] N. Khan and S. Anandaraj, "A Survey on Preserving Data Confidentiality in Cloud Computing Using Different Schemes", *Advances in Intelligent Systems and Computing*, pp. 211-219, 2021. Available: 10.1007/978-981-16-1249-7_21.

[22] "XChaCha20-Poly1305 construction – Libsodium A modern, portable, easy to use crypto library.", 2021. [Online]. Available: https://doc.libsodium.org/secret-key_cryptography/aead/chacha20-poly1305/xchacha20-poly1305_construction.

[23] "Stack Overflow Developer Survey 2021", *Stack Overflow*, 2021. [Online]. Available: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-programming-scripting-and-markup-languages. [Accessed: 13- Dec- 2021].

[24] F. Thabit, S. Alhomdy and S. Jagtap, "Security analysis and performance evaluation of a new lightweight cryptographic algorithm for cloud computing", *Global Transitions Proceedings*, vol. 2, no. 1, pp. 100-110, 2021. Available: 10.1016/j.gltp.2021.01.014.

[25] J. Aumasson, S. Fischer, S. Khazaei, W. Meier and C. Rechberger, "New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba", *Fast Software Encryption*, pp. 470-488. Available: 10.1007/978-3-540-71039-4_30.

[26] B. Jungk and S. Bhasin, "Don't fall into a trap: Physical side-channel analysis of ChaCha20-Poly1305", *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017. Available: 10.23919/date.2017.7927155

[27] H. Böck, A. Zauner, S. Devlin, J. Somorovsky and P. Jovanovic, "Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS", 2021.

[28] "Speeding up and strengthening HTTPS connections for Chrome on Android", *Google Online Security Blog*, 2021. [Online]. Available: https://security.googleblog.com/2014/04/speeding-up-and-strengthening-https.html. [Accessed: 14- Dec- 2021].

[29] N. Sullivan, "Do the ChaCha: better mobile performance with cryptography", 2021. [Online]. Available: https://blog.cloudflare.com/do-the-chacha-better-mobile-performance-with-cryptography/. [Accessed: 14- Dec- 2021].

[30] "NordPass. 2021. XChaCha20 Encryption", *Nordpass.com*, 2021. [Online]. Available: https://nordpass.com/features/xchacha20-encryption/. [Accessed: 14- Dec- 2021].

[31] D. Koo, J. Hur and H. Yoon, "Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage", *Computers & Electrical Engineering*, vol. 39, no. 1, pp. 34-46, 2013. Available: 10.1016/j.compeleceng.2012.11.002.

[32] D. Bernstein, "ChaCha, a variant of Salsa20", 2021.

[33] M. Darwich, Y. Ismail, T. Darwich and M. Bayoumi, "Cost-Efficient Storage for On-Demand Video Streaming on Cloud", *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020. Available: 10.1109/wf-iot48130.2020.9221374.

[34] P. Crowley and E. Biggers, "Adiantum: length-preserving encryption for entry-level processors", *IACR Transactions on Symmetric Cryptology*, pp. 39-61, 2018. Available: 10.46586/tosc.v2018.i4.39-61.

[35] “Report: 18.1% of all Files Uploaded to the Cloud Contain Sensitive data", *McAfee Blogs*, 2021. [Online]. Available: https://www.mcafee.com/blogs/enterprise/cloud-security/report-18-1-of-all-files-uploaded-to-the-cloud-contain-sensitive-data/. [Accessed: 14- Dec- 2021].

[36] "Ransomware Maze", *McAfee Blogs*, 2021. [Online]. Available: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/ransomware-maze/. [Accessed: 14- Dec- 2021].

[37] S. Maitra, "Chosen IV cryptanalysis on reduced round ChaCha and Salsa", *Discrete Applied Mathematics*, vol. 208, pp. 88-97, 2016. Available: 10.1016/j.dam.2016.02.020 [Accessed 14 December 2021].

[38] S. Arciszewski, "XChaCha: eXtended-nonce ChaCha and AEAD_XChaCha20_Poly1305 ["work in progress"]", *Tools.ietf.org*, 2021. [Online]. Available: https://tools.ietf.org/id/draft-irtf-cfrg-xchacha-03.html. [Accessed: 14- Dec- 2021].

[39] "How to Protect Data at Rest with Amazon EC2 Instance Store Encryption | Amazon Web Services", *Amazon Web Services*, 2021. [Online]. Available: https://aws.amazon.com/blogs/security/how-to-protect-data-at-rest-with-amazon-ec2-instance-store-encryption/. [Accessed: 14- Dec- 2021].

[40] A. Mari, "Brazilian Ministry of Health hit by second cyberattack in less than a week | ZDNet", *ZDNet*, 2021. [Online]. Available: https://www.zdnet.com/article/brazilian-ministry-of-health-hit-by-second-cyberattack-in-less-than-a-week/. [Accessed: 14- Dec- 2021].

[41] "Trusted Platform Module (TPM) fundamentals", *Docs.microsoft.com*, 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/security/information-protection/tpm/tpm-fundamentals. [Accessed: 15- Dec- 2021].

[42] T. Edgar and D. Manz, *Research Methods for Cyber Security*. Syngress, 2017, p. 49.

[43] J. Rosenberg, "Embedded security", *Rugged Embedded Systems*, p. e72, 2017. Available: 10.1016/b978-0-12-802459-1.00011-7.