PLOS ONE

RESEARCH ARTICLE

Ensuring privacy and confidentiality of cloud data: A comparative analysis of diverse cryptographic solutions based on run time trend

John Kwao Dawson[1,2]*, **Twum Frimpong**[2], **James Benjamin Hayfron Acquah**[2], **Yaw Marfo Missah**[2]

1 Sunyani Technical University, Sunyani, Ghana, **2** Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

* kwaodawson@stu.edu.gh

OPEN ACCESS

Citation: Dawson JK, Frimpong T, Hayfron Acquah JB, Missah YM (2023) Ensuring privacy and confidentiality of cloud data: A comparative analysis of diverse cryptographic solutions based on run time trend. PLoS ONE 18(9): e0290831. https://doi.org/10.1371/journal.pone.0290831

Editor: Pandi Vijayakumar, University College of Engineering Tindivanam, INDIA

Received: April 11, 2023

Accepted: August 16, 2023

Published: September 7, 2023

Peer Review History: PLOS recognizes the benefits of transparency in the peer review process; therefore, we enable the publication of all of the content of peer review and author responses alongside final, published articles. The editorial history of this article is available here: https://doi.org/10.1371/journal.pone.0290831

Data Availability Statement: https://www.kaggle.com/datasets/morriswongch/kaggle-datasets.

Funding: The authors received no specific funding for this work.

Abstract

The cloud is becoming a hub for sensitive data as technology develops, making it increasingly vulnerable, especially as more people get access. Data should be protected and secured since a larger number of individuals utilize the cloud for a variety of purposes. Confidentiality and privacy of data is attained through the use of cryptographic techniques. While each cryptographic method completes the same objective, they all employ different amounts of CPU, memory, throughput, encryption, and decryption times. It is necessary to contrast the various possibilities in order to choose the optimal cryptographic algorithm. An integrated data size of $5n{*}10^2$ (KB ($\in$ 1,2,4,10,20,40) is evaluated in this article. Performance metrics including run time, memory use, and throughput time were used in the comparison. To determine the effectiveness of each cryptographic technique, the data sizes were run fifteen (15) times, and the mean simulation results were then reported. In terms of run time trend, NCS is superior to the other algorithms according to Friedman's test and Bonferroni's Post Hoc test.

1.0 Introduction

Human activity has risen, making communication more difficult, and necessitating data protection. [1]. A paradigm shift in data storage needs to be implemented in order to secure these enormous amount of data [2]. Due to the enormous amount of data produced by numerous social media platforms, including Facebook, Twitter, Instagram, and e-commerce websites, cloud computing is currently the preferred option [3].

Amazon's four-hour cloud computing downtime in 2017 cost S&P 500 Company $150 million, according to a Maeser [4]. A network traffic control organization called Apica predicted that the top 54 e-commerce sites will experience a decline in activity of at least 20% [4]. According to Ponemon, Fortune 1000 companies lost just over $ 2.5 billion in 2015 as a result of data center shutdowns brought on by hackers. According to Maeser [4], the need for cloud computing will increase by about 266% between 2013 and 2020 as a result of the massive volumes of data that the Internet of Things will produce.

Competing interests: The authors have declared that no competing interests exist.

Once more, Maeser [4] stressed that the infrastructure-as-a-service aspect of cloud computing will result in an increase in demand of roughly 85%.

Because of the benefits of agility, scalability, availability, accelerating the development of work, and lowering operating costs by utilizing pay-as-you-use services, cloud computing continues to gain popularity over traditional on-site data centers [5, 6]. As a result, Information Technology giants are now investing far more money on cloud computing than they did in the past. These advantages have led businesses to use cloud services such as Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS), Platform-as-a-Service, and Container-as-a-Service (CaaS) for their pay-as-you-use activities on the cloud [7–9].

The adoption of cloud computing is accompanied by a number of security issues, such as data privacy and confidentiality [10–12]. In order to guarantee the secrecy and privacy of cloud data, cryptographic techniques have shown to be an effective and efficient methods [12–16, 18].

In this study, the symmetric algorithms Enhanced RSA (ERSA) [17], Non-Deterministic Cryptographic Scheme (NCS) [18], Enhanced Homomorphic Scheme (EHS) [19], Chacaha20, and Salsa20 are compared based on run time trend, throughput time, and memory usage to determine which algorithm among them could be used to ensure the confidentiality and privacy of cloud data.

1.1 Identified problem

The biggest problem with cloud computing has been data security. Researchers have suggested many encryption technique variations to protect cloud data [20]. Enhanced RSA [17], Non-Deterministic Cryptographic Scheme [18], Enhanced Homomorphism Scheme [19], Chacha20, and Salsa20 are a few examples of these techniques. These suggested encryption techniques are effective in limiting unauthorized access to sensitive information.

In terms of run time trend, throughput time, and memory complexity, it is unclear which one performs better than the other. Such knowledge is essential for industry startups, other professionals, and researchers who are interested in utilizing effective algorithms to protect privacy and confidentiality of data in the cloud. Therefore, the major goal of this research is to test Enhanced RSA, Non-Deterministic Cryptographic Schemes, Enhanced Homomorphism Schemes, Chacha20, and Salsa20 in order to determine the computational statistics of the best method. Once more, this study offers a solid framework that theoretically and practically combines all of the recognized algorithms into a robust system. The principal contribution of this paper is the proposition of a comprehensive cryptographic scheme(s) that can be used to ensure confidentiality and privacy of cloud data.

2.0 Literature review

Cloud computing users save their data in the cloud making it a remote location based system. It compromises secrecy, which is essential for cloud computing to be acceptable. To boost security and trustworthiness, cryptography is often used in cloud computing.

ALmarwani et al. [21] presented a unique tagging approach called Tagging of Outsourced Data (TOD) in an endeavor to protect the secrecy of data stored in the cloud. Their method supported cloud data through verification. Their method had a short run time, enabling for widespread use by mobile devices. Tahir et al. [22] presented a genetic algorithm called CryptoGa to help Almarwani et al. achieve data privacy. When compared to state-of-the-art algorithms like AES, RSA, and DES, their approach had shorter execution times.

Shen et al. [23] advocated using proxy re-encryption and Oblivious Random Access Memory (ORAM). Their technique was designed to ensure multi-user data sharing on the cloud.

The ciphertext gained through the proxy re-encryption enabled members to regulate access and, as a consequence, establish data privacy.

Garad et al. [24] suggested a cryptosystem to protect submitted files to the cloud server. AES-CCM, AES-GCM, and CHACHA20_POLY1305 were the asymmetric cryptographic algorithms they employed. They divide the file into N pieces, then use various cryptographic techniques to encrypt each portion. Thabit, et al. [25], ensured the confidentiality and privacy of cloud data by proposing a Lightweight Cryptographic Algorithm. Their algorithm integrated Feistel and substitution schemes to raise the encryption complexity of their algorithm. Their algorithm was very effective regarding run time. Tiwari and Neogi [26] proposed a security scheme that secured a multi-tenant hybrid cloud by combining Kerberos Authentication Protocol with Resource Allocation Manager Unit (RAMU). Their scheme allowed for more resource access while also improving client confidentiality and security. The model validates the user's request before providing access, preventing the password from being revealed to hackers during transmission. The Key Distribution Centre (KDC) validates the request and RAMU grants access after reviewing the control database and resource allocation map. Gadde et al. [27] suggested an Improved Blowfish cryptography strategy for encrypting and decrypting sensitive data in the cloud server using the optimum key. Optimal key creation is a critical method for achieving integrity and confidentiality goals of integrity and confidentiality. Similarly, data restoration is the inverse process of sanitization (decryption). In [28] a four-step data security approach in cloud computing was proposed. They used the least significant bit of LSB approaches to integrate three cryptographic algorithms, RSA, AES, and identity-based encryption, with steganography to attain confidentiality and privacy of cloud data.

3.0 Methodology

3.1 Enhanced RSA

Enhanced RSA Improved traditional RSA's security by combining classic RSA with Gaussian interpolation formula. The integration raises the security of RSA to the fifth level. After encrypting the message's ASCII values with Gaussian First Forward interpolation, the conventional RSA is used to encrypt and decode the message at the second and third levels. The last stage uses Gaussian First Backward interpolation to decode the data again, as seen in Fig 1. The integration helps to overcome the classic RSA factorization problem [17].

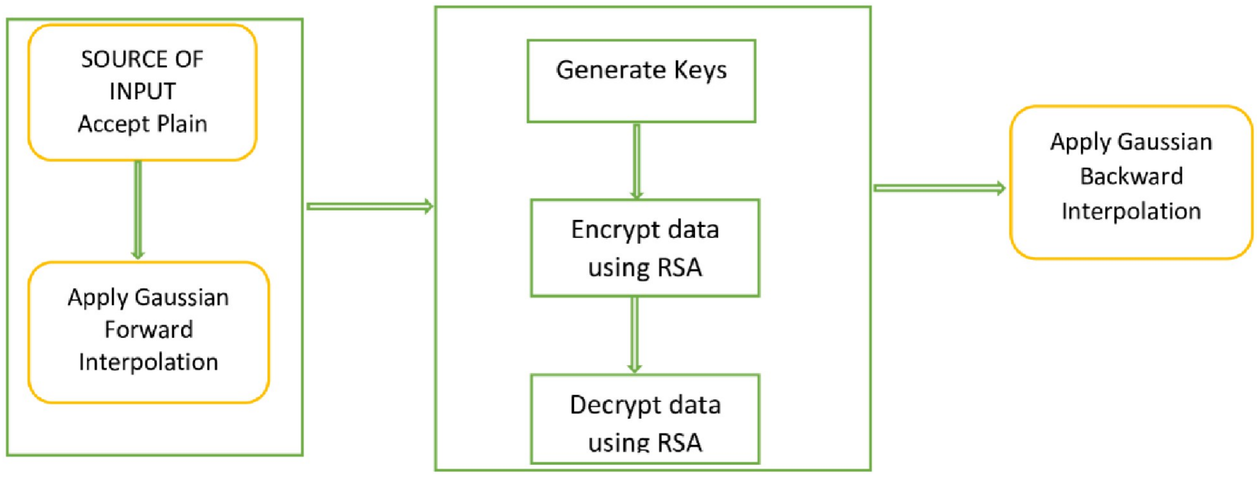

SOURCE OF INPUT Accept Plain
Apply Gaussian Forward Interpolation
Generate Keys
Encrypt data using RSA
Decrypt data using RSA
Apply Gaussian Backward Interpolation

Fig 1. Work flow diagram of enhanced RSA [17].

https://doi.org/10.1371/journal.pone.0290831.g001

3.2 Non-Deterministic Cryptographic Scheme

This method consists of three stages: key generation, encryption, and decryption. The three levels of key production are meant to produce secret keys that help to secure the algorithm [18]. These include the use of Good Prime numbers, the Linear Congruential Generator, the Fixed Sliding Window Algorithm, and XORing the output with the plaintext. In NCS a sub array of $\left(\frac{n(a[i])}{4}\right)$ is computed on the twelve numbers generated after the application of the Fixed Sliding Window [18].

3.3 Enhanced Homomorphism Scheme

Enhanced Homomorphism Scheme (EHS) is developed and implemented by the amalgamation of Good Prime Numbers (GPN), Linear Congruential Generator (LCG), Fixed Sliding Window Algorithm (FSWA), and Gentry's Algorithm. Two stages are considered in this algorithm, the generation of keys and the application of the homomorphism scheme. Three procedures are used to generate the keys which includes the generation of two good prime numbers with the product as a seed for the Linear Congruential Generator to produce twelve numbers. The sliding window algorithm is applied on the twelve numbers using a sub-array of three $\left(\frac{n(a[i])}{3}\right)$. The first value is s_i, second value s_j, the third value s_k, fourth value is s_l and with M the plaintext as seen in Eq 1 for data encryption [19].

$$C = M + s_i * s_j + s_k * s_l \tag{1}$$

3.4 Salsa20

The Salsa20 stream cipher, initially developed by Bernstein in 2008, has 20 rounds, while it's more recent variations, Salsa20/8 and Salsa20/12, have 8 and 12 rounds respectively. Salsa20/20 refers to the 20-round Salsa20. The Salsa20 stream cipher accepts keys of 128 and 256 bits. The Salsa20 core is composed of a 256-bit key (n0, n1, n2, n3, n4, n5, n6, n7), a 64-bit block counter (e0, e1), a 64-bit nonce (d0, d1), and four 32-bit diagonal constants (c0, c1, c2, c3) that may be mapped into a 4 4 matrix as in Eq 2.

$$X = \begin{pmatrix} b_0^0 & b_1^0 & b_2^0 & b_3^0 \\ b_4^0 & b_5^0 & b_6^0 & b_7^0 \\ b_8^0 & b_9^0 & b_{10}^0 & b_{11}^0 \\ b_{12}^0 & b_{13}^0 & b_{14}^0 & b_{15}^0 \end{pmatrix} \rightarrow \begin{pmatrix} c_0 & n_0 & n_1 & n_2 \\ n_3 & c_1 & d_0 & d_1 \\ e_0 & e_1 & c_2 & n_4 \\ n_5 & n_6 & n_7 & c_3 \end{pmatrix} \tag{2}$$

The three operations of Salsa20, addition, rotation, and XOR, give desired cryptographic features, with modular addition providing non-linearity and bit rotation providing diffusion in a word. The diffusion attribute is propagated from one word to the next via the XOR technique [29].

3.5 Chacha20

Chacha20 is a 2008 update of Salsa20 that uses a new round function to boost diffusion. Salsa20 uses a 32-bit module addition, XOR, and rotation operations based core hash function, whereas Chacha20 uses an internal state of sixteen 32-bit words arranged as a 4 × 4 matrix to map a 256-bit key (128-bit key is also suitable for Salsa20), a 64-bit nonce, and a 64-bit counter to a 512-bit block key stream. While the configurations differ, their beginning states are both made up of 8 words of key, 2 words of stream position (counter), 2 words of nonce, and 4 words of constant. Chacha20 is a replacement for Salsa20's quarterround, which updates each word twice with the same amount of operations as shown in Eq 3. Unlike Salsa20, which alternates quarterrounds down columns and across rows, Chacha20 executes quarterrounds down columns and along diagonals in a doubleround. Chacha20 likewise employs 10 double-round iterations and the same output function [30].

$$\left\{\begin{array}{lll} k+=x; & z\oplus=k; & z\lll<=16; \\ y+=z; & x\oplus=y; & x\lll<=12; \\ k+=x; & z\oplus=\ k; & z\lll<=8; \\ y+=z; & x\oplus=y; & x\lll<=7; \end{array}\right\} \quad (3)$$

3.6 The proposed framework of the system

This section provides a broad summary of the comparative study of the enhanced RSA, Non-Deterministic Cryptographic Scheme, enhanced Homomorphism Scheme, Chacha20, and Salsa20. For the ERSA, NCS, EHS, ChaCa20, and Salsa20, the architecture is divided into five phases: key generation, encryption, decryption, memory utilization, and throughput. From Fig 2, the user registers with a cloud service provider. The registered client uploads the plain-text onto the cloud after encryption using either ERSA, NCS, EHS, Chaca20, or Salsa20 to obtain Ciphertext from which Encryption time, memory usage, and throughput time are computed. This is achieved using any digital device [6]. The decryption time, memory storage

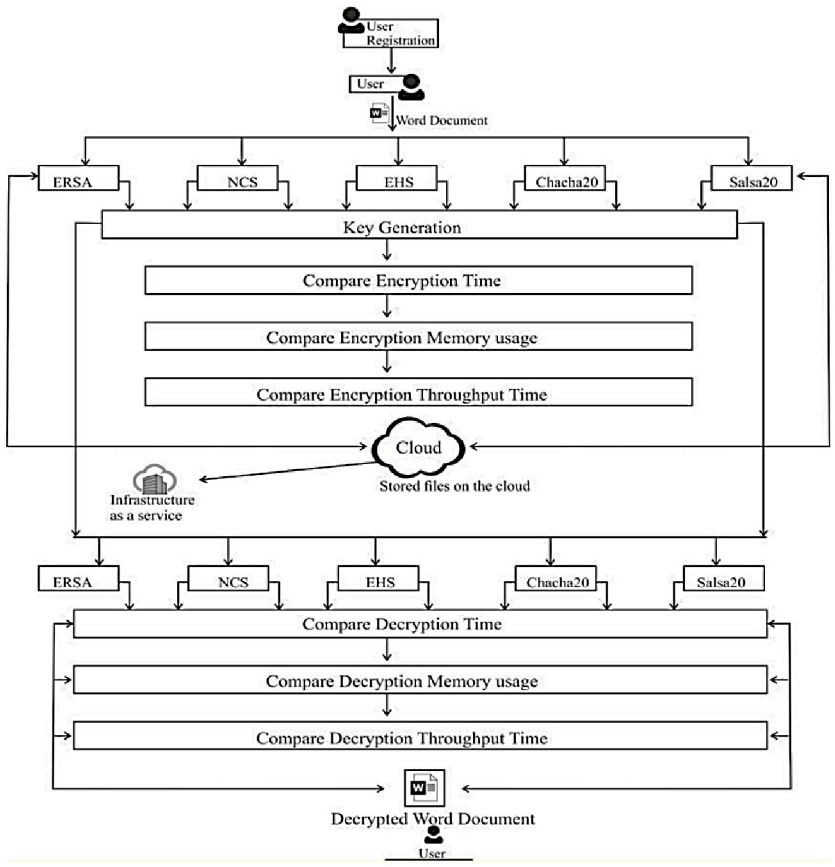
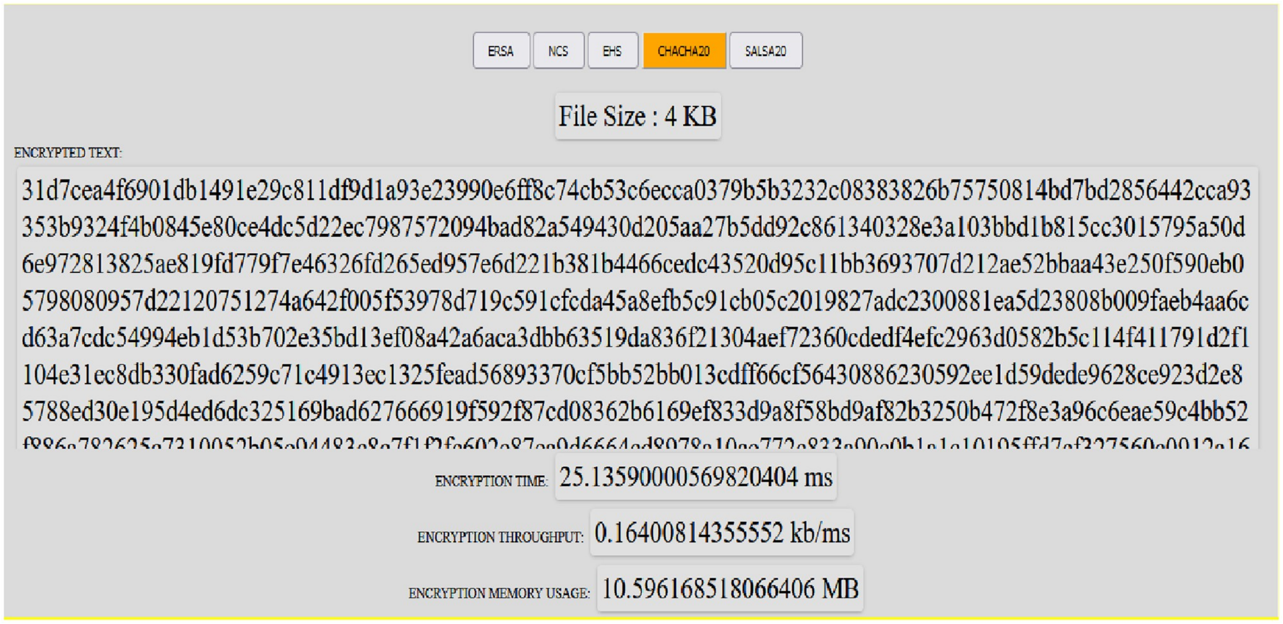

User Registration
User
Word Document
ERSA
NCS
EHS
Chacha20
Salsa20
Key Generation
Compare Encryption Time
Compare Encryption Memory usage
Compare Encryption Throughput Time
Cloud
Stored files on the cloud
Infrastructure as a service
ERSA
NCS
EHS
Chacha20
Salsa20
Compare Decryption Time
Compare Decryption Memory usage
Compare Decryption Throughput Time
Decrypted Word Document
User

Fig 2. Framework for the proposed model.

https://doi.org/10.1371/journal.pone.0290831.g002



File Size : 4 KB

ENCRYPTED TEXT:

```
31d7cea4f6901db1491e29c811df9d1a93e23990e6ff8c74cb53c6ecca0379b5b3232c08383826b75750814bd7bd2856442cca93
353b9324f4b0845e80ce4dc5d22ec7987572094bad82a549430d205aa27b5dd92c861340328e3a103bbd1b815cc3015795a50d
6e972813825ae819fd779f7e46326fd265ed957e6d221b381b4466cedc43520d95c11bb3693707d212ae52bbaa43e250f590eb0
5798080957d22120751274a642f005f53978d719c591cfcda45a8efb5c91cb05c2019827adc2300881ea5d23808b009faeb4aa6c
d63a7cdc54994eb1d53b702e35bd13ef08a42a6aca3dbb63519da836f21304aef72360cdedf4efc2963d0582b5c114f411791d2f1
104e31ec8db330fad6259c71c4913ec1325fead56893370cf5bb52bb013cdff66cf56430886230592ee1d59dede9628ce923d2e8
5788ed30e195d4ed6dc325169bad627666919f592f87cd08362b6169ef833d9a8f58bd9af82b3250b472f8e3a96c6eae59c4bb52
```

ENCRYPTION TIME: 25.13590000569820404 ms

ENCRYPTION THROUGHPUT: 0.16400814355552 kb/ms

ENCRYPTION MEMORY USAGE: 10.596168518066406 MB

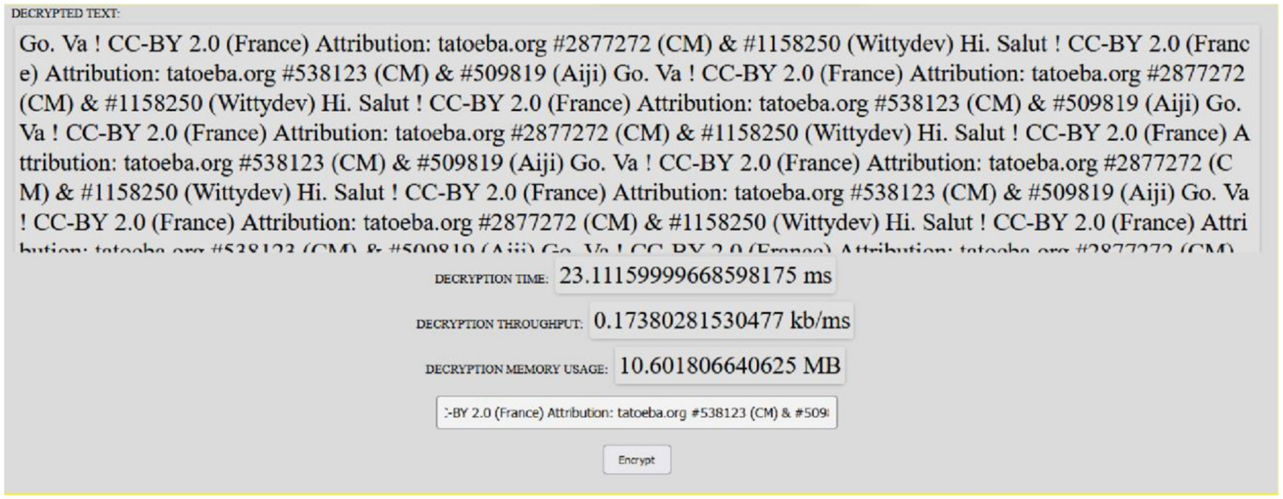
Fig 3. Encrypted performance for Chacha20 using 4KB data.

https://doi.org/10.1371/journal.pone.0290831.g003

time, and throughput time are calculated once the Ciphertext has been downloaded. Figs 3 and 4 show a snapshot of the run time for 4KB data for Chacha20 using the system, with further information available at https://github.com/Elkie1/Chacha20.

4.0 Experimentation

The comparative analysis of the Enhanced RSA [17], Non-Deterministic Cryptographic Scheme [18], Enhanced Homomorphism Scheme [19], Salsa20 and Chacha20 was implemented on an i7 Lenovo computer, 2.10 GHz CPU using C# language. C# programming



DECRYPTED TEXT:

Go. Va ! CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev) Hi. Salut ! CC-BY 2.0 (Franc e) Attribution: tatoeba.org #538123 (CM) & #509819 (Aiji) Go. Va ! CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev) Hi. Salut ! CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #509819 (Aiji) Go. Va ! CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev) Hi. Salut ! CC-BY 2.0 (France) A ttribution: tatoeba.org #538123 (CM) & #509819 (Aiji) Go. Va ! CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (C M) & #1158250 (Wittydev) Hi. Salut ! CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #509819 (Aiji) Go. Va ! CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev) Hi. Salut ! CC-BY 2.0 (France) Attri

DECRYPTION TIME: 23.11159999668598175 ms

DECRYPTION THROUGHPUT: 0.17380281530477 kb/ms

DECRYPTION MEMORY USAGE: 10.601806640625 MB

:-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #509

Fig 4. Decrypted performance for Chacha20 using 4KB data.

https://doi.org/10.1371/journal.pone.0290831.g004

language is preferable because it influences the execution time of data as was implemented in a NET C# programming language where it was used to test the execution of AES algorithm resulting in 300MB/seconds while OpenSSL C simulation produced a 960 MB/seconds average speed [31].

4.1 Description of dataset used in this work

The study's dataset was collected from the Kaggle database [32]. The dataset was used to assess the robustness of the algorithms in terms of run time trend, memory use, and throughput. The dataset is an English-to-French translation that includes text, numbers, and special characters. This is critical since Loyka et al.'s [33] investigation showed divergent results when only text and numbers were utilized. The proposed algorithms were evaluated using data sizes of $5n*10^2$ (KB ($\in$ 1,2,4,10,20,40). The dataset was run fifteen (15) times to ensure the accuracy of the run time, and the mean and standard deviation of the execution were computed.

5.0 Results

5.1 Encryption time

From Table 1, Chacha20 had the lowest mean encryption time of 15.9047±1.69 milliseconds followed by NCS (52.13±31.1766) with Salsa20 having the highest mean encryption time of 853±85.06 milliseconds when data size of 500KB (0.5MB) was executed. The encryption time increased from 853±85.06 milliseconds to 1302.8 ±703.97 milliseconds for Salsa20 when the data size was increased to 1000KB making it linear [34, 35]. The encryption time for ERSA also increased from 462.93±40.93 milliseconds to 575.67±57.05 millisecond when data size was raised to 1000KB. However, the encryption time for NCS reduced from 147.33±172.41 milliseconds to 85.8±54.46 milliseconds and to 82.2±75.17 milliseconds when data size was increased from 500KB to 1000KB, and from 2000KB to 5000KB.

5.2 Decryption time

Table 2, presents the comparison of the mean decryption time trend for Salsa20, Chacha20, ERSA, NCS, and EHS. With data size of 500KB, NCS had the lowest decryption time of 74 ±45.16 milliseconds followed by Chacha20 (281.33±35.42) and EHS with a decryption time of 368.4±133.88 milliseconds. When the data size was increased to 1000KB, NCS had the lowest decryption time of 105.6±71.78 milliseconds with Salsa20 having the highest run time of 869.4 ±223.18. From Table 2, it could be seen that for Salsa20, Chacha20 and ERSA their run time increases as data sizes increases making them linear (O (N)). However, the run time for NCS and EHS alternates as the sizes of the data increases making it non-linear.

Table 1. Comparing the mean encryption time for Salsa20, Chacha20, ERSA, NCS, and EHS.

Data Size(KB)	Salsa20 (ms)	Chacha20(ms)	ERSA (ms) [17]	NCS (ms) [18]	EHS (ms) [19]
500	853±85.06	15.9047±1.69	462.93±40.93	52.13±31.1766	178.27±64.57
1000	1302.8 ±703.97	19.8653±1.65	575.67±57.05	147.33±172.41	204.33±132.99
2000	2936.2±777.81	195.5867±32.21	695.93±71.59	85.8±54.46	268.93±158.19
5000	5916.33±764.99	4186.7373±1343.42	738±150.31	82.2±75.17	139.27±75.45
10000	11536.27±2860.06	5995.722±1932.96	866.67±84.74	98.07±121.57	118.67±39.85
20000	32531.07±46652.36	7769.3793±1169.29	1679.4±2514.36	163.07±168.07	148.53±95.19

https://doi.org/10.1371/journal.pone.0290831.t001

Table 2. Comparing the mean and standard deviation decryption time for Salsa20, Chacha20, ERSA, NCS, and EHS.

Data Size (KB)	Salsa20 (ms)	Chacha20(ms)	ERSA (ms) [17]	NCS (ms) [18]	EHS (ms) [19]
500	869.4±223.18	281.33±35.42	391.4±47.08	74±45.16	368.4±133.88
1000	2152.13±2469.76	508±73.48	514.2±75.87	105.6±71.78	277.4±144.83
2000	3020.41±1369.13	1524.93±2466.48	612.4±89.93	159.53±108.66	234.8±105.27
5000	5888.8±864.34	6138.27±1792.07	732.93±94.29	151.07±155.33	253.13±168.95
10000	10582.13±1274.53	10028.53±2757.07	1375.47±121.33	115.93±123.58	315.53±122.98
20000	19699.33±1068.30	15893.53±1934.37	1874.73±3528.41	158.87±144.66	137.87±71.83

https://doi.org/10.1371/journal.pone.0290831.t002

5.3 Encryption throughput (KB/ms)

The number of units of data that can be processed at a given time is considered throughput [36]. This is computed using Eq 4.

$$Throughput = \frac{Size\ of\ Data}{Run\ Time} \tag{4}$$

From Table 3, with 500KB of data, Salsa20 had the lowest mean encryption throughput time of 0.039078 KB/ms with Chacha20 having the highest throughput time of 31.43731 KB/ms. Also, when the data size was increased to 2000KB, NCS had the highest mean encryption throughput time of 23.31002 KB/ms with Salsa20 having the lowest mean encryption throughput time of 0.04541KB/ms followed by ERSA with a mean encryption throughput time of 2.87383 KB/ms. However, when data size was increased to 10000KB, NCS had the highest encryption throughput 101.9714 KB/ms. However, with a data size of 20000KB, EHS had the highest mean encryption throughput time of 134.6499102 KB/ms followed by NCS with a mean encryption throughput time of 122.6492 KB/ms.

5.4 Decryption throughput

From Table 4, NCS had the highest mean decryption throughput time of 6.756757 KB/ms with Salsa20 having the lowest mean decryption throughput time of 0.575109 KB/ms when data size of 500KB was executed. Again when the data size was increased to 2000KB, NCS had the highest mean decryption throughput of 12.53656 KB/ms followed by EHS with a mean decryption throughput time of 8.517887564 KB/ms. However, with a data size of 20000KB, EHS had the highest mean decryption throughput time of 145.0676983 KB/ms followed by NCS with a mean decryption throughput time of 125.8917 KB/ms.

Table 3. Comparing the mean encryption throughput (KB/ms) for Salsa20, Chacha20, ERSA, NCS, and EHS.

Data Size (KB)	Salsa20(KB/ms)	Chacha20(KB/ms)	ERSA (KB/ms) [17]	NCS (KB/ms) [18]	EHS (KB/ms) [19]
500	0.039078	31.43731	1.080069	9.590793	2.804786836
1000	0.051172	50.33895	1.737116	6.78733	4.893964111
2000	0.04541	10.22565	2.873838	23.31002	7.436787308
5000	0.056341	1.194247	6.775068	60.82725	35.90234562
10000	0.057789	1.667856	11.53846	101.9714	84.26966292
20000	0.040986	2.574208	11.90902	122.6492	134.6499102

https://doi.org/10.1371/journal.pone.0290831.t003

Table 4. Comparing the mean decryption throughput (KB/ms) for Salsa20, Chacha20, ERSA, NCS, and EHS.

Data Size (KB)	Salsa20(KB/ms)	Chacha20(KB/ms)	ERSA (KB/ms) [17]	NCS (KB/ms) [18]	EHS(KB/ms)[19]
500	0.575109	1.777251	1.277466	6.756757	1.357220413
1000	0.464655	1.968504	1.944769	9.469697	3.604902668
2000	0.662161	1.311533	3.265839	12.53656	8.517887564
5000	0.849069	0.814562	6.821903	33.09797	19.75243613
10000	0.9449	0.997155	7.27062	86.25647	31.6923727
20000	1.015263	1.258373	10.66818	125.8917	145.0676983

https://doi.org/10.1371/journal.pone.0290831.t004

5.5 Encryption memory usage

From Table 5, with a data size of 500KB, Salsa20 has high memory complexity of 196.4±32.24 megabytes followed by Chcaha20 (191.8±35.91 megabytes) with EHS having the lowest memory complexity of 16±2.75 megabytes. When data size was increased to 20000KB, Salsa20 used 6160.27±650.17 megabytes of memory which was still the highest followed by Chacha20 (6092.27±653.58) while NCS used the least memory complexity of 16.4±3.78 megabytes.

5.6 Decryption memory usage

From Table 6, Salsa20 has the highest mean memory complexity of 192.13±34.41 megabytes of memory when 500KB of data was decrypted with NCS having the lowest mean memory complexity of 15.87±3.04 megabytes. However, when data size was increased to 20000 KB, Chacha20 had the highest memory complexity of 6281.4±713.08 megabytes with NCS still having the lowest memory complexity of 17.07±3.28 megabytes.

5.7 Comparing significance difference between the encryption and decryption times using Friedman Test and Bonferroni Post Hoc test

The Friedman's Test tests the hypothesis that "all treatment effects are zero" as against the alternate hypothesis "not all treatment effects are zero". From the output in Tables 7 and 9,

Table 5. Comparing the mean encryption memory usage for Salsa20, Chacha20, ERSA, NCS, and EHS (MB).

Data Size (KB)	Salsa20 (MB)	Chacha20(MB)	ERSA (MB) [17]	NCS (MB) [18]	EHS (MB) [19]
500	196.4±32.24	191.8±35.91	20.6±2.59	17.13±1.99	16±2.75
1000	270.33±34.78	265.13±38.54	20.6±2.47	16.13±3.87	16.13±3.64
2000	588.6±66.89	597.67±66.57	20.6±2.13	16.33±3.75	16.53±3.25
5000	1443.2±192.69	1392.33±224.02	20.13±2.07	16.13±3.98	16.47±3.62
10000	3436.13±355.35	3453.13±365.96	21.33±2.09	15.87±3.44	17.4±3.62
20000	6160.27±650.17	6092.27±653.58	20.73±2.46	16.4±3.78	17.6±3.12

https://doi.org/10.1371/journal.pone.0290831.t005

Table 6. Comparing the mean decryption memory usage for Salsa20, Chacha20, ERSA, NCS, and EHS (MB).

Data Size (KB)	Salsa20(MB)	Chacha20(MB)	ERSA (MB) [17]	NCS (MB) [18]	EHS (MB) [19]
500	192.13±34.41	191.87±35.89	20.67±2.16	15.87±3.04	16.2±3.05
1000	270±34.78	259.47±40.63	20.23±2.37	16.13±3.87	17.5±3.74
2000	595.13±68.94	587.47±78.23	20.6±2.32	16.47±3.6	17.6±3.15
5000	1360.53±236.62	1384.53±241.7	20.87±2.13	16.33±3.79	17.6±3.64
10000	3453.73±368.56	3404.87±374.45	20.53±2.19	16.2±3.14	17.7±3.13
20000	6267.13±732.39	6281.4±713.08	21.2±2.83	17.07±3.28	18.2±2.68

https://doi.org/10.1371/journal.pone.0290831.t006

Table 7. Friedman test of the encryption times for Chacha20, Salsa20, ERSA, NCS and EHS.

Null hypothesis		H_0: All treatment effects are zero	
Alternative hypothesis		**H_1: Not all treatment effects are zero**	
Method	**DF**	**Chi-Square**	**P-Value**
Not adjusted for ties	4	22.03	0.000
Adjusted for ties	4	22.22	0.000

https://doi.org/10.1371/journal.pone.0290831.t007

P-Value = 0.00 < 0.05(alpha value), which indicates that the difference in the encryption times is statistically significant for the different algorithms and data sizes.

A Post Hoc Bonferroni pairwise comparison test was used to see if there was a significant difference between pairs of algorithms and data sizes. The encryption timings for EHS and NCS were statistically different from Salsa20, Chacha20, and ERSA with P-Values less than 0.05 ($P - Value < 0.05, Reject\ H_0$) according to Table 8.

From Table 10, the decryption Times for Salsa20—Chacha20, ERSA -EHS, and NCS -EHS are statistically not different.

6.0 Discussions

From Table 1, it could be deduced that the encryption times for Salsa20 and Chacha20 were proportional to the data sizes executed which resulted from the addictive, XORing, and constant distance rotation during execution [37, 38]. ERSA encryption times also showed a proportional relationship between data size and encryption time [17]. This made their encryption times predictable, deterministic, and patterned which confirms the work of Masram et al. [39] and [38, 40–45]. The use of longer keys ensures higher security but results in higher CPU utilization when encryption time is dependent on data size (O (N)) [46]. However, the use of smaller keys is the best employed in cloud computing due to less CPU engagement [47].

The encryption time for NCS and EHS is non-patterned, non-deterministic, and unpredictable because of the disintegration of the keys through the application of a Fixed Sliding Window Algorithm and XORing of the keys and the plaintext which makes NCS and EHS resistant to breaking the resultant cipher through XORing any captured encoded text [48]. Again the randomization from the application of the Sliding Window Algorithm helps to increase the security of the encrypted data and also reduces the time complexity of the Non-Deterministic

Table 8. Bonferroni simultaneous tests for differences of means of encryption time.

Difference of Algorithm Levels	Difference of Means	SE of Difference	Simultaneous 95% CI	T-Value	Adjusted P-Value
EHS—CHACHA20	-248.98	7.83	(-271.08, -226.89)	-31.80	0.000
ERSA—CHACHA20	-167.97	7.83	(-190.07, -145.88)	-21.45	0.000
NCS—CHACHA20	-257.23	7.83	(-279.32, -235.13)	-32.85	0.000
SALSA20—CHACHA20	0.54	7.83	(-21.55, 22.64)	0.07	1.000
ERSA—EHS	81.01	7.83	(58.92, 103.11)	10.35	0.000
NCS—EHS	-8.24	7.83	(-30.34, 13.85)	-1.05	1.000
SALSA20—EHS	249.53	7.83	(227.43, 271.62)	31.87	0.000
NCS—ERSA	-89.26	7.83	(-111.35, -67.16)	-11.40	0.000
SALSA20—ERSA	168.52	7.83	(146.42, 190.61)	21.52	0.000
SALSA20—NCS	257.77	7.83	(235.68, 279.87)	32.92	0.000

Individual confidence level = 99.50%

https://doi.org/10.1371/journal.pone.0290831.t008

Cryptographic Scheme [49, 50]. From the trend of the encryption time from Table 1, it could be concluded that encryption times for NCS and EHS are not dependent on data size but the size of the key while the encryption times for Salsa20, Chacha20, and ERSA are influenced by data size.

From these discussions, it could be concluded that data size is proportional to the decryption time for Salsa20, Chaca20, and ERSA as indicated in Table 2. This makes their trend of decryption time deterministic, predictable, and linear which is supported by the works of [44, 51]. With a linear trend of decryption time, hackers can predict, intercept and modify data [52].

Based on these discussions, it is possible to conclude that ERSA, Salsa20, and Chacha20 produced linear, predictable, deterministic, and high decryption times, making them vulnerable to side-channel attacks and thus do not guarantee the absolute privacy and confidentiality of data on the cloud, as suggested by Kumar et al. [34] and Karthik [35]. The application of the Fixed Sliding Window Algorithm, which disintegrates the huge numbers obtained from the selection of the good prime numbers as the initial keys, Linear Congruential Generator, and XORing the keys and the Ciphertext to obtain plaintext, caused NCS and EHS to have unpredictable, non-deterministic, and non-linear decryption time.

This has the advantage of reducing bandwidth utilization since data encryption and decryption raise the overhead cost of data processing [53]. Again, non-linear encryption timings aided in increasing data secrecy and privacy while reducing device ripping and wear for industry participants and people [18].

According to Tables 1 and 3, encryption time is inversely related to throughput time. Algorithms with faster throughput times use less CPU, and vice versa [54].

When Tables 2 and 4 were compared, it was possible to deduce that with a long decryption time, the associated throughput time was short. This supports the findings of Abolade et al. [54], who discovered that algorithms with a high throughput time need less CPU time.

Table 5 shows that algorithms that consume less memory during execution serve to decrease computational bottlenecks for the CPU and, as a consequence, are regarded as the best [55]. According to Table 6, Salsa20 is more memory intensive since its operation is based on 20 cycles with 10 repeating instances [56]. Because the secret key and Ciphertext are XORed without padding, the NCS had the lowest mean memory usage.

The Friedman Test and Bonferroni Post Hoc test results from Tables 7–10 show that the encryption and decryption times for NCS and EHS are statistically different from Salsa20, Chacha20, and ERSA.

It could be summarized that NCS and EHS produced lower, non-deterministic, non-patterned, and secret key-dependent run times as such defeats the idea behind ERSA, Salsa20, and Chacha20 as the fastest symmetric algorithms which used less memory during data execution [49]. This makes NCS a lightweight algorithm to be employed in the cloud and other areas where fast and lightweight algorithms are needed. Also, it could be used in environments where mobile devices and other devices with less memory are used such as the Internet of Things.

Table 9. Friedman test of the decryption times for Chacha20, Salsa20, ERSA, NCS and EHS.

Null hypothesis		H_0: All treatment effects are zero
Alternative hypothesis		H_1: Not all treatment effects are zero
DF	Chi-Square	P-Value
4	22.93	0.000

https://doi.org/10.1371/journal.pone.0290831.t009

Table 10. Bonferroni simultaneous tests for differences of means for decryption time.

Difference of Algorithm Levels	Difference of Means	SE of Difference	Simultaneous 95% CI	T-Value	Adjusted P-Value
EHS—CHACHA20	-2000.8	32.8	(-2093.3, -1908.3)	-61.01	0.000
ERSA—CHACHA20	-1997.6	32.8	(-2090.1, -1905.0)	-60.92	0.000
NCS—CHACHA20	-2001.9	32.8	(-2094.5, -1909.4)	-61.05	0.000
SALSA20—CHACHA20	4.8	32.8	(-87.7, 97.4)	0.15	1.000
ERSA—EHS	3.2	32.8	(-89.3, 95.7)	0.10	1.000
NCS—EHS	-1.1	32.8	(-93.7, 91.4)	-0.03	1.000
SALSA20—EHS	2005.6	32.8	(1913.1, 2098.2)	61.16	0.000
NCS—ERSA	-4.3	32.8	(-96.9, 88.2)	-0.13	1.000
SALSA20—ERSA	2002.4	32.8	(1909.9, 2095.0)	61.06	0.000
SALSA20—NCS	2006.8	32.8	(1914.2, 2099.3)	61.20	0.000

Individual confidence level = 99.50%

https://doi.org/10.1371/journal.pone.0290831.t010

7.0 Conclusion

To ensure the secrecy and privacy of data in the cloud, modern cryptographic techniques are applied to encode and decode data. These encryption techniques have computational overheads that have an impact on cloud performance. The symmetric stream cipher algorithms; ERSA, Salsa20, Chacha20, NCS, and EHS have all been thoroughly examined. For securing the privacy and secrecy of data stored in the cloud, ERSA, Salsa20, and Chacha20 are seen to be strong cryptographic schemes. However, compared to NCS and EHS, their run times are linear, predictable, and long, rendering them vulnerable to side-channel attacks.

Their linear runtime trends result in significant bandwidth use and hardware device wear and tear during the transfer of large amounts of data, making them unsuitable for a cloud computing environment. Additionally, because of their linear run times, hackers can estimate the execution time of any piece of data. The Friedman test and Bonferroni Post Hoc test, however, showed that NCS and EHS had the advantage of producing non-linear run time trends, non-patterned run time trends, non-deterministic run time trends, lowest run times, high throughput, and consumed less amount of memory during execution.

Since NCS and EHS will ensure reduced bandwidth usage, prevent tearing and wearing of hardware, and maximize the utilization of any device without much attention on the specification of hardware, this provides industry players and academia optimism that they can fully embrace cloud computing. Future studies should focus on doing experiments using computers with greater specifications. Additionally, research should be done to compare the security strength of NCS to other cutting-edge algorithms.

Author Contributions

Conceptualization: John Kwao Dawson.

Data curation: John Kwao Dawson.

Formal analysis: John Kwao Dawson.

Investigation: John Kwao Dawson.

Methodology: John Kwao Dawson.

Supervision: Twum Frimpong, James Benjamin Hayfron Acquah, Yaw Marfo Missah.

References

1. M. Tajammul, R. Parveen and I. A. Tayubi, "Comparative Analysis of Security Algorithms used in Cloud Computing," *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2021, pp. 875–880.
2. Ibrahim I. A. and Bassiouni M., "Improvement of job completion time in data-intensive cloud computing applications," *Journal of Cloud Computing*, vol. 9, no. 1, Feb. 2020, https://doi.org/10.1186/s13677-019-0139-6
3. Seth B. et al., "Secure Cloud Data Storage System using Hybrid Paillier-Blowfish Algorithm," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 779–798, 2021, https://doi.org/10.32604/cmc.2021.014466
4. Maeser R., "Analyzing CSP Trustworthiness and Predicting Cloud Service Performance," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 73–85, 2020, https://doi.org/10.1109/OJCS.2020.2994095 PMID: 32406827
5. Hassan J. et al., "The Rise of Cloud Computing: Data Protection, Privacy, and Open Research Challenges-A Systematic Literature Review (SLR)," *Computational intelligence and neuroscience*, vol. 2022, p. 8303504, Jun. 2022, https://doi.org/10.1155/2022/8303504 PMID: 35712069
6. Kim Y.-K., Kim H.-J., Lee H., and Chang J.-W., "Privacy-preserving parallel kNN classification algorithm using index-based filtering in cloud computing," *PLOS ONE*, vol. 17, no. 5, p. e0267908, May 2022, https://doi.org/10.1371/journal.pone.0267908 PMID: 35511912
7. Mustafa M., Alshare M., Bhargava D., Neware R., Singh B., and Ngulube P., "Perceived Security Risk Based on Moderating Factors for Blockchain Technology Applications in Cloud Storage to Achieve Secure Healthcare Systems," *Computational and Mathematical Methods in Medicine*, vol. 2022, pp. 1–10, Jan. 2022, https://doi.org/10.1155/2022/6112815 PMID: 35096132
8. Al-Issa Y., Ottom M. A., and Tamrawi A., "eHealth Cloud Security Challenges: A Survey," *Journal of Healthcare Engineering*, vol. 2019, pp. 1–15, Sep. 2019, https://doi.org/10.1155/2019/7516035 PMID: 31565209
9. Hussein M., Mousa M., and Alqarni M., "A placement architecture for a container as a service (CaaS) in a cloud environment", *Journal of Cloud Computing*, vol. 8, no. 1, 2019. Available: https://doi.org/10.1186/s13677-019-0131-1
10. P. Kulkarni, R. Khanai, and G. Bindagi, "A Comparative Analysis of Hybrid Encryption Technique for Images in the Cloud Environment," *2020 International Conference on Communication and Signal Processing (ICCSP)*, Jul. 2020.
11. R. K. Nema, A. K. Saxena and R. Srivastava, "Survey of the Security Algorithms over Cloud Environment to Protect Information," 2022 10th International Conference on Emerging Trends in Engineering and Technology—Signal and Information Processing (ICETET-SIP-22), 2022, pp. 1–6.
12. Mittal S. et al., "Using Identity-Based Cryptography as a Foundation for an Effective and Secure Cloud Model for E-Health," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–8, Apr. 2022, https://doi.org/10.1155/2022/7016554 PMID: 35510050
13. Kumar P., Suresh A., Anbarasu V., Anandaraj S. P., and Udayakumar S., "A decentralized secured grid integration system using APEBC technique with multi-access AI framework," *Sustainable Computing: Informatics and Systems*, vol. 35, p. 100777, Sep. 2022, https://doi.org/10.1016/j.suscom.2022.100777
14. Abu-Faraj M., Al-Hyari A., Aldebei K., Alqadi Z. A. and Al-Ahmad B., "Rotation Left Digits to Enhance the Security Level of Message Blocks Cryptography," in IEEE Access, vol. 10, pp. 69388–69397, 2022, https://doi.org/10.1109/ACCESS.2022.3187317
15. Chinnasamy P. and Deepalakshmi P., "HCAC-EHR: hybrid cryptographic access control for secure EHR retrieval in healthcare cloud," *Journal of Ambient Intelligence and Humanized Computing*, Feb. 2021, https://doi.org/10.1007/s12652-021-02942-2
16. Mangla C., Rani S., and Atiglah H. K., "Secure Data Transmission Using Quantum Cryptography in Fog Computing," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–8, Jan. 2022, https://doi.org/10.1155/2022/3426811
17. Dawson J. K., Twum F., Acquah J. B. H., Missah Y. M., and Ayawli B. B. K., "An enhanced RSA algorithm using Gaussian interpolation formula," *International Journal of Computer Aided Engineering and Technology*, vol. 16, no. 4, p. 534, 2022, https://doi.org/10.1504/ijcaet.2022.123996
18. Dawson J. K., Twum F., Acquah J. H., and Missah Y. M.," Ensuring Confidentiality and Privacy of Cloud Data Using a Non-Deterministic Cryptographic Scheme," Ploseone, https://doi.org/10.1371/journal.pone.0274628 PMID: 36758028
19. Dawson J. K., Twum F., Acquah J. H., and Missah Y. M., "Ensuring privacy and confidentiality of data on the cloud using an enhanced homomorphism scheme," *Informatica*, vol. 46, no. 8, Nov. 2022, https://doi.org/10.31449/inf.v46i8.4305

20. Liu L., Cao M., and Sun Y., "A fusion data security protection scheme for sensitive E-documents in the open network environment," *PLOS ONE*, vol. 16, no. 12, p. e0258464, Dec. 2021, https://doi.org/10.1371/journal.pone.0258464 PMID: 34910722

21. ALmarwani R., Zhang N., and Garside J., "An effective, secure and efficient tagging method for integrity protection of outsourced data in a public cloud storage," *PLOS ONE*, vol. 15, no. 11, p. e0241236, Nov. 2020, https://doi.org/10.1371/journal.pone.0241236 PMID: 33151960

22. Tahir M., Sardaraz M., Mehmood Z., and Muhammad S., "CryptoGA: a cryptosystem based on genetic algorithm for cloud data security," *Cluster Computing*, Jul. 2020, https://doi.org/10.1007/s10586-020-03157-4

23. Shen J., Yang H., Vijayakumar P., and Kumar N., "A Privacy-Preserving and Untraceable Group Data Sharing Scheme in Cloud Computing," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, pp. 2198–2210, 1 July-Aug. 2022, https://doi.org/10.1109/TDSC.2021.3050517

24. Garad A., Agrawal R., Suryawanshi A., and Rodagi J., "Securing File Storage in Cloud Using Hybrid," International Journal of Advances in Engineering Research. 23 (5), 44–50, 2022.

25. Thabit F., Alhomdy A. P. S., Al-Ahdal A. H. A., and Jagtap P. D. S., "A new lightweight cryptographic algorithm for enhancing data security in cloud computing," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 91–99, Jun. 2021, https://doi.org/10.1016/j.gltp.2021.01.013

26. Tiwari Shiv Kumar and Neogi Subhrendu Guha, "Design and Implementation of Enhanced Security Algorithm for Hybrid Cloud using Kerberos," vol. 4, no. 5, Jun. 2023, https://doi.org/10.1007/s42979-023-01807-z

27. Gadde S., Amutharaj J, and Usha S., "Cloud Multimedia Data Security by Optimization-Assisted Cryptographic Technique," Jan. 2023, https://doi.org/10.1142/s0219467824500104

28. Abdulraheem W., "Performance Comparison of Xen AND Hyper-V in Cloud Computing While Using Cryptosystems," vol. 14, no. 3, pp. 18–30, Nov. 2022, https://doi.org/10.15849/ijasca.221128.02

29. Mazumdar B., Ali Sk. S., and Sinanoglu O., "A Compact Implementation of Salsa20 and Its Power Analysis Vulnerabilities," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, pp. 1–26, Dec. 2016, https://doi.org/10.1145/2934677

30. Jiao L., Hao Y., and Feng D., "Stream cipher designs: a review," *Science China Information Sciences*, vol. 63, no. 3, Feb. 2020, https://doi.org/10.1007/s11432-018-9929-x

31. M. H. Alrowaithy, "Performance-efficient cryptographic primitives in constrained devices," *theses.ncl.ac.uk*, 2021. http://theses.ncl.ac.UK/jspui/handle/10443/5545 (accessed Oct. 22, 2022).

32. "Kaggle Datasets", *Kaggle.com*, 2022. [Online]. https://www.kaggle.com/datasets/morriswongch/kaggle-datasets. [Accessed: 27- Sep- 2022].

33.] K. Loyka, H. Zhou, and S. P. Khatri, "A Homomorphic Encryption Scheme Based on Affine Transforms," *Proceedings of 2018 on Great Lakes Symposium on VLSI*, May 2018.

34. Kumar S. et al., "Side Channel Attack on Stream Ciphers: A Three-Step Approach To State/Key Recovery," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 166–191, Feb. 2022, https://doi.org/10.46586/tches.v2022.i2.166-191

35. S. Karthik, "Introduction to Timing Attacks!," *Medium*, Dec. 23, 2020. https://medium.com/spidernitt/introduction-to-timing-attacks-4e1e8c84b32b

36. Liu W., Feng W., Huang M., Xu Y., and Zheng X., "STEB: A secure service trading ecosystem based on blockchain," *PLOS ONE*, vol. 17, no. 6, p. e0267914, Jun. 2022, https://doi.org/10.1371/journal.pone.0267914 PMID: 35657907

37. Serrano R., Duran C., Sarmiento M., Pham C.-K., and Hoang T.-T., "Chacha20–Poly1305 Authenticated Encryption with Additional Data for Transport Layer Security 1.3," *Cryptography*, vol. 6, no. 2, p. 30, Jun. 2022, https://doi.org/10.3390/cryptography6020030

38. M. Panda and A. Nag, "Plain Text Encryption Using AES, DES, and SALSA20 by Java Based Bouncy Castle API on Windows and Linux," 2015 Second International Conference on Advances in Computing and Communication Engineering, 2015, pp. 541–548.

39. Masram R., Shahare V., Abraham J., and Moona R., "Analysis and Comparison of Symmetric Key Cryptographic Algorithms Based on Various File Features," *International Journal of Network Security & Its Applications*, vol. 6, no. 4, pp. 43–52, Jul. 2014, https://doi.org/10.5121/ijnsa.2014.6404

40. Banerjee A., Hasan M., Rahman Md. A., and Chapagain R., "CLOAK: A Stream Cipher Based Encryption Protocol for Mobile Cloud Computing," *IEEE Access*, vol. 5, pp. 17678–17691, 2017, https://doi.org/10.1109/access.2017.2744670

41. Abd Zaid M. and Hassan S., "Proposal Framework to Light Weight Cryptography Primitives," *Engineering and Technology Journal*, vol. 40, no. 4, pp. 516–526, Apr. 2022, https://doi.org/10.30684/etj.v40i4.1679

42. Alyas H. H. and Abdullah A. A., "Enhancement the Chacha20 Encryption Algorithm Based on Chaotic Maps," *Lecture Notes in Networks and Systems*, pp. 91–107, 2021.

43. P. Yadav, I. Gupta, and S. K. Murthy, "Study and analysis of eSTREAM cipher Salsa and ChaCha," 2016 IEEE International Conference on Engineering and Technology (ICETECH), 2016, pp. 90–94.

44. Salkanovic A., Ljubic S., Stankovic L., and Lerga J., "Analysis of Cryptography Algorithms Implemented in Android Mobile Application," *Information Technology and Control*, vol. 50, no. 4, pp. 786–807, Dec. 2021, https://doi.org/10.5755/j01.itc.50.4.29464

45. P. Singh and K. Deshpande, "Performance evaluation of cryptographic ciphers on IoT devices," 2018. [Online]. https://arxiv.org/ftp/arxiv/papers/1812/1812.02220.pdf

46. Sun Q., Lin K., Si C., Xu Y., Li S. and Gope P., "A Secure and Anonymous Communicate Scheme over the Internet of Things", *ACM Transactions on Sensor Networks*, vol. 18, no. 3, pp. 1–21, 2022. Available: https://doi.org/10.1145/3508392 [Accessed 28 September 2022].

47. Papaioannou M. et al., "A Survey on Security Threats and Countermeasures in Internet of Medical Things (IoMT)", *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 6, 2020. Available: https://doi.org/10.1002/ett.4049 [Accessed 28 September 2022].

48. Stone W., Kim D., Kemmoe V. Y., Kang M., and Son J., "Rethinking the Weakness of Stream Ciphers and Its Application to Encrypted Malware Detection," *IEEE Access*, vol. 8, pp. 191602–191616, 2020, https://doi.org/10.1109/access.2020.3030559

49. R. Alsharida, M. Hammood, M. A. Ahmed, B. Thamer, and M. Shakir, "RC4D: A New Development of RC4 Encryption Algorithm," *Selected Papers from the 12th International Networking Conference*, pp. 19–30, 2021.

50. Mojisola F. O., Misra S., Falayi Febisola C., Abayomi-Alli O., and Sengul G., "An improved random bit-stuffing technique with a modified RSA algorithm for resisting attacks in information security (RBMRSA)," *Egyptian Informatics Journal*, vol. 23, no. 2, pp. 291–301, Jul. 2022, https://doi.org/10.1016/j.eij.2022.02.001

51. Noor N. S., Hammood D. A., Al-Naji A., and Chahl J., "A Fast Text-to-Image Encryption-Decryption Algorithm for Secure Network Communication," *Computers*, vol. 11, no. 3, p. 39, Mar. 2022, https://doi.org/10.3390/computers11030039

52. Srinivasan A. B., S H., and mika R., "A Table-Based End to End Encryption Technique Without Key Exchange," *Engineered Science*, 2022, https://doi.org/10.30919/es8d705

53. Gui R., Yang L., and Gui X., "An Order-Preserving Encryption Scheme Based on Weighted Random Interval Division for Ciphertext Comparison in Wearable Systems," *Sensors*, vol. 22, no. 20, p. 7950, Oct. 2022, https://doi.org/10.3390/s22207950 PMID: 36298301

54. Abolade O., Okandeji A., Oke A., Osifeko M., and Oyedeji A., "Overhead effects of data encryption on TCP throughput across IPSEC secured network," *Scientific African*, vol. 13, p. e00855, Sep. 2021, https://doi.org/10.1016/j.sciaf.2021.e00855

55. Pi D., Liu J., and Wang Y., "Review of computer-generated hologram algorithms for color dynamic holographic three-dimensional display," *Light: Science & Applications*, vol. 11, no. 1, p. 231, Jul. 2022, https://doi.org/10.1038/s41377-022-00916-3 PMID: 35879287

56. Lara E., Aguilar L., García J., and Sanchez M., "A Lightweight Cipher Based on Salsa20 for Resource-Constrained IoT Devices," *Sensors*, vol. 18, no. 10, p. 3326, Oct. 2018, https://doi.org/10.3390/s18103326 PMID: 30287774