# Chaos-based Encryption Keys and Neural Key-store for Cloud-hosted Data Confidentiality

N.N Mosola[1,2], M.T Dlamini[3,4], J.M Blackledge[2], J.H.P Eloff[3], H.S Venter[3]

[1]*Department of Mathematics and Computer Science, National University of Lesotho, Lesotho*

[1]nn.mosola@nul.ls

[2] *School of Mathematics, Statistics and Computer Science, University of Kwa-Zulu Natal, Durban, South Africa*

[2]216075642@student.ukzn.ac.za

blackledge@ukzn.ac.za

[3]*Department of Computer Science, University of Pretoria, Pretoria, South Africa*

[3]{mdlamini; eloff; hventer}@cs.up.ac.za

[4] *Command, Control and Information Warfare, Defence, Peace, Safety and Security, Council of Scientific and Industrial Research, Pretoria, South Africa*

[4]*TDlamini1@csir.co.za*

*Abstract*— **Cloud computing brings flexible and cost effective services. However, security concerns plague the cloud. Data confidentiality is one of the concerns inhibiting the adoption of cloud computing. This concern stems from various cyberattacks directed towards gaining unauthorised access to cloud-bound or cloud-hosted data. This paper proposes a client-end encryption and key management system to curb attacks that targets compromising the confidentiality of cloud-hosted data. The proposed system uses chaotic atmospheric noise to generate a fitness function. The fitness function generates random numbers which create encryption keys. The strength of the encryption keys is derived from the chaotic and random nature of the atmospheric noise. The keys are then used for encrypting cloud-bound data using Advanced Encryption Standard (AES-128, 192 and 256), Data Encryption Standard (DES), 3-DES, and our novel cryptosystem named Cryptor, before it can be sent to the cloud. However, encryption bears no significance if the key management is flawed. To address the inherent key management problem, the solution uses a neural network to learn patterns of an encryption key. Once learnt, the key is then discard to thwart possible key attacks. The key is reconstructed by the neural network for decryption purposes.**

***Keywords***— **Cloud computing, confidentiality, chaotic noise, encryption, neural network.**

## I. INTRODUCTION

Cloud computing has gained considerable popularity. This is due to the exponential increase in the use of the Internet-based services e.g. software as a service (SaaS), platform as a service (PaaS), infrastructure as a service (IaaS). Cloud computing brings attractive benefits. For example, resource sharing, storage capacities, pay-per-use model etc. However, despite having such advantages, the cloud is plagued with security concerns. Among the concerns is data confidentiality breaches.

Quite often, information security practitioners make use of encryption systems to achieve data confidentiality guarantees. Several encryption systems are used to secure cloud services. For example, Data Encryption Standard (DES), its variant triple DES, Advanced Encryption Standard (AES), the Rivest, Adleman and Shamir (RSA) algorithm and others. However, cybercriminals are continually finding new ways of compromising the confidentiality of cloud-hosted data and services. Cybercriminals continue to develop new tools, techniques and procedures (TTPs) to breach existing encryption systems and steal cloud-hosted data. Hence, the adoption of cloud-based services hinges on getting right the issues that relate to data confidentiality [1]. Data confidentiality issues mainly arise from the fact that cloud-based services can be accessed from virtually anywhere, at any time, and using any Internet-enabled device. Such convenience in terms of accessibility opens gaping vulnerabilities that get exploited by various attacks that target compromising the confidentiality of cloud-hosted data. For example, inter-VM, VM-sprawl and insider attacks exploit vulnerabilities in virtual machines and hypervisors to breach the confidentiality of cloud-hosted data. Existing cryptographic solutions are proving to be insufficient in dealing with these new threats. This has created a need for better mechanism to deal with the new threats that are specific to the cloud. Therefore, new encryption systems are required to provide data confidentiality guarantees and to thwart these new cyber-attacks.

Moreover, due to the emergence of various digital devices such as smart phones, tablets, laptops etc., enormous amounts of digital content are generated and sent for storage on the cloud. Cloud service providers (CSPs) such as Google, Dropbox, Microsoft, Apple, Amazon etc., offer individuals, small and big organizations cloud storage services for them to store their data. This service comes at a fraction of the cost of hosting the data in-house. Thus, cloud computing is bulging with digital content. The digital content explosion is envisaged to increase exponentially in years to come. The increasing use of cloud storage services to store digital content calls for secure encryption measures to provide strong confidentiality guarantees on the data. Therefore, some CSPs already provide encryption mechanisms to ensure secure storage services that guarantee confidentiality to cloud-hosted data. For example, Dropbox uses AES-256 and a transport layer security (TLS) protocol to provide data confidentiality for its customers [2]. Despite the efforts of some CSPs in trying to implement strong encryption to protect cloud-hosted data from any malicious attacks, there have been numerous security breaches that have resulted in confidential data leakages. For example, the same Dropbox which uses AES-256 has experienced a cyber-attack in which user credentials were stolen and used to reveal customers' cloud-hosted data [3].

Furthermore, some CSPs are bound by service level agreements (SLAs), regulatory compliance and legal policies to provide confidentiality guarantees on cloud-hosted data [4]. But still data leakage breaches are widespread and come with huge consequences. Surely, there is something wrong with the current approach. It appears that most encryption systems are implemented on the CSP's end which indicates that the resultant encryption keys are also managed by the CSP. However, some CSPs have moved toward encrypting customers' cloud-hosted data and hand the keys back to the customers (i.e. the keys to the data are not on the CSPs premises). Some CSPs have involved third parties to manage encryption keys. All these approaches have been tried and tested. However, it must be noted that weak encryption or a flawed key management systems often result in confidential data leakages which has severe financial implications to the customer and the breached CSP. Thus, encryption on the cloud requires careful attention and has to be done right in order to provide the right levels of confidentiality guarantees to cloud-hosted data.

Therefore, this paper addresses data confidentiality issues on the cloud from the perspective of strong encryption and secure key management. The remainder of this paper is structured as follows: section II discusses existing literature. Section III introduces the proposed model. Section IV presents and discusses the results of the proposed solution. Section V concludes the paper and provides future recommendations.

## II. LITERATURE REVIEW

Various research related to cloud encryption has been conducted. In the quest to achieve data confidentiality, encryption is cited as the widely-used method [5][6].

Neural cryptography is a new technique for providing data confidentiality [7]. This technique combines the concepts of machine learning and cryptography, using neural networks. In their research, [7] proposes artificial intelligence techniques to invent cryptosystems to curb eavesdropping. The research proposes two artificial neural networks for develop a cryptographic algorithm to protect data. The encryption part was a success as the neural networks successfully communicated securely, avoiding eavesdropping. However, the solution requires large memory due to the exponential growth of the neural networks. It also takes longer periods to exchange a secret key between the two communicating neural networks. These issues make the solution inefficient.

A neural cryptographic scheme is proposed by [8]. This scheme is based on mutual machine learning concepts. The idea of mutual machine learning is gaining popularity in various aspects of neural cryptography. For example, using synchronization, the prevalent key distribution problems faced by most encryption systems might be solved. The proposed scheme uses two feed-forward neural networks (NNs). The NNs have discrete and continuous weights. The proposed scheme encrypts data successfully. However, the work of [9] shows that the solution proposed by [8] is susceptible to genetic, geometric and probabilistic attacks. Using genetic algorithms, the scheme can be easily defeated by mutating a fitness function until a matching encryption key is found. This is analogous to a brute-force attack.

A study conducted by [9] proposes a neural cryptographic scheme using information substitution and permutations. This two techniques are meant to achieve confusion and diffusion when encrypting data. This scheme uses a recursive, modulo-2 substitution and two feed forward NNs. Communicating NNs receive a unique input vector to produce a unique output bit. This process is repeated a number of times to generate a secret key. This scheme encrypts plaintext through a recursive modulo-2 substitution phase to produce ciphertext. This ciphertext is then enciphered to produce the final ciphertext, using a cipher block chain (CBC) and an exclusive OR (i.e. XOR) operation. The CBC and *XOR* operations are applied on vectors with identical weights and intermediate ciphertext block lengths. As with most neural network implementations, neural cryptographic schemes require large memory as the neural network grows exponentially and requires huge amounts of training time.

Homomorphic encryption is one of the methods proposed for ensuring data confidentiality. The notion of homomorphic encryption was introduced by Rivest, Adleman and Dertouzous [10]. Homomorphic encryption is based on the ability to perform certain computations, such as addition and multiplication on ciphertext, without using a decryption key to decrypt the data. This allows third parties such as CSPs, to perform limited queries on ciphertext while preserving the confidentiality of the data. However, [11] shows that homomorphic encryption schemes have vulnerabilities and can be broken. In particular, a study by [12] proved that given a deterministic, privacy homomorphic scheme can be broken in sub-exponential time. This becomes worse in a quantum computing space. A study by [13] showed that homomorphic scheme with deterministic properties can be broken using quantum computations. Furthermore, existing schemes are not fully homomorphic as they allow only one operation, either addition or multiplication, on a ciphertext. A fully homomorphic encryption has been proposed by [14][15]. However, fully homomorphic encryption schemes are slow, resource intensive and has not been tested for practical implementations.

Another study on encryption, key management and data confidentiality on the cloud was conducted by [16]. This research proposes a client-end cryptosystem for encrypting data prior to uploading it to the cloud. Although the cryptosystem achieves data confidentiality, its major downfall is on the key management system. The authors propose to blend the encryption key with the ciphertext. This becomes security by obscurity - a vulnerability as cybercriminals might find out that the key is part of the ciphertext. Hence, the confidentiality of the data may be compromised.

A study by [17] uses a light-weight AES-128, a secure hash algorithm (SHA-512) and water-marking to provide confidentiality of cloud-hosted data. This study also uses the Bell & Lapadula model for authorization purposes. The authors implement a real-time identification scheme to curb data leaks through a cache technique. The technique they implement uses a VM cache memory to identify authenticated and authorized users. The contents of the VM cache memory are encrypted with AES-128 and sent to an authentication server. The authentication server decrypts the message header to find the user credentials. However, virtual cache memory may not uniquely identify cached data due to aliasing. Aliasing means that a virtual memory address may be mapped to a different physical memory address. Hence, virtual cache memory is

susceptible to end channel attacks as revealed in [18]. The authors show how a novel attack vector can easily exploit vulnerabilities of the hypervisor and other software security monitors. The attack vector uses aliasing to place incoherent copies of physical addresses on a cache memory. Thus, virtual indexing is a one-to-many function. Such functions may not be desirable where unique identifiers are used, such as in a cloud environment. Hence, an unauthorized user might gain access to confidential data.

An approach combining encryption and data fragmentation is proposed by [19][20]. Fragmentation ensures that data is split into several fragments which can be stored in distributed cloud databases provided by CSPs. Fragmentation may be implemented either vertically or horizontally across a relational database to store data in various data centers. This is also very useful for backup purposes. However, efficiency in data retrieval may be an expensive process especially if one of the sites (i.e. database storing fragments) is inaccessible due to network problems as the fragments will not be enough to reconstruct the data.

To ensure confidentiality guarantees, this study advocates for client-end encryption. This approach aims to avoid confidential data leaks due to intentional or accidental incidents. CSPs may also be compelled to provide "back doors" for law enforcement agencies to have uninterrupted access and surveillance of cloud-hosted data. For example, the widely-reported case between the Federal Bureau of Investigation (FBI) and Apple, the Microsoft court case on stored emails etc., have set new standards on future cryptographic systems. Testimony to this is the quick move by WhatsApp to develop an end-to-end encryption scheme for their instant messaging application.

This paper extends the work of [20], which proposes a neural cryptosystem for cloud-bound data. This work shows how cloud users can encrypt their confidential data before uploading it to the cloud. The cryptosystem proposes a one-time pad (OTP) in which the data to be encrypted must be of equal length to the encryption key. This study [29] further proposes the use of a counter propagation neural network for key storage and revocation. However, the proposed system is not tested against existing encryption solutions to determine its efficiency. Hence this paper adopts the techniques of [29] to generate encryption keys for DES, 3-DES and AES to have an objective analysis of the encryption systems.

In summary, various researchers have suggested several methods of protecting cloud-hosted data through encryption and key management. However, existing systems fall short when it comes to providing data confidentiality guarantees. Existing research suggests solutions requiring high processing power and large memory requirements. This becomes a problem when considering client-end security. Another issue with existing cryptosystems is their increasing reliance on third partyKDCs. The idea of KDCs requires a lot of trust as the third-party entity can have access to the confidential data. Moreover, KDCs are often targeted by cybercriminals. Hence, new encryption schemes that provide data confidentiality guarantees and secure encryption key storage are needed. Therefore, this paper proposes a light-weight chaos-based encryption system. It also aims to strengthen existing encryption schemes by producing encryption keys from chaotic random noise in the quest to have strong keys for DES, 3-DES and AES. The next section discusses the proposed model.

## III. PROPOSED MODEL SOLUTION

This section introduces the proposed solution which aims to fill the gaps identified in the reviewed literature and provide confidentiality guarantees through a client-end encryption scheme. Encryption and key management is a big issue, especially in a multi-tenant and distributed environment like the cloud. The proposed model seeks to address the issue of data confidentiality through strong encryption. The proposed scheme is based on symmetric key encryption. This means that the same key is used to encrypt and decrypt data. Furthermore, the proposed scheme is based on evolutionary computing concepts. These concepts introduce a paradigm shift in terms of conventional approaches to encrypting data associated with the Kerckhoff-Shannon principle. The concepts adhere to static algorithms and dynamic encryption keys.

Moreover, evolutionary computing concepts explore how encryption algorithms and keys can quite literally be generated 'on the fly'. This is done so that a user can be provided with, or, better still, individually generate a personalised encryption algorithm in addition to a personalized encryption key.

The model has the following functional requirements. Each of these requirements is taken into consideration in the design of the proposed model.

### *A. Functional requirements*

- Client-end: The system must be able to encrypt cloud user's data before it could be uploaded on the cloud. The encryption keys and everything concerning the model should be done by the client. The CSP must never gain access to the encryption algorithm or keys.
- Light-weight: The system must be able to encrypt and decrypt data on the fly and without requiring a lot of computational resources. This is to enable our proposed solution to be suitable for devices with minimal computing resources like mobile devices.
- Secure key management – The system must be able to manage encryption keys securely and effectively. This also must be done on the client end.
- Self-destruction of keys – The system must be designed such that it can discard encryption keys once the encryption is complete.

Fig.1 below depicts the architecture of the proposed encryption and key management scheme.

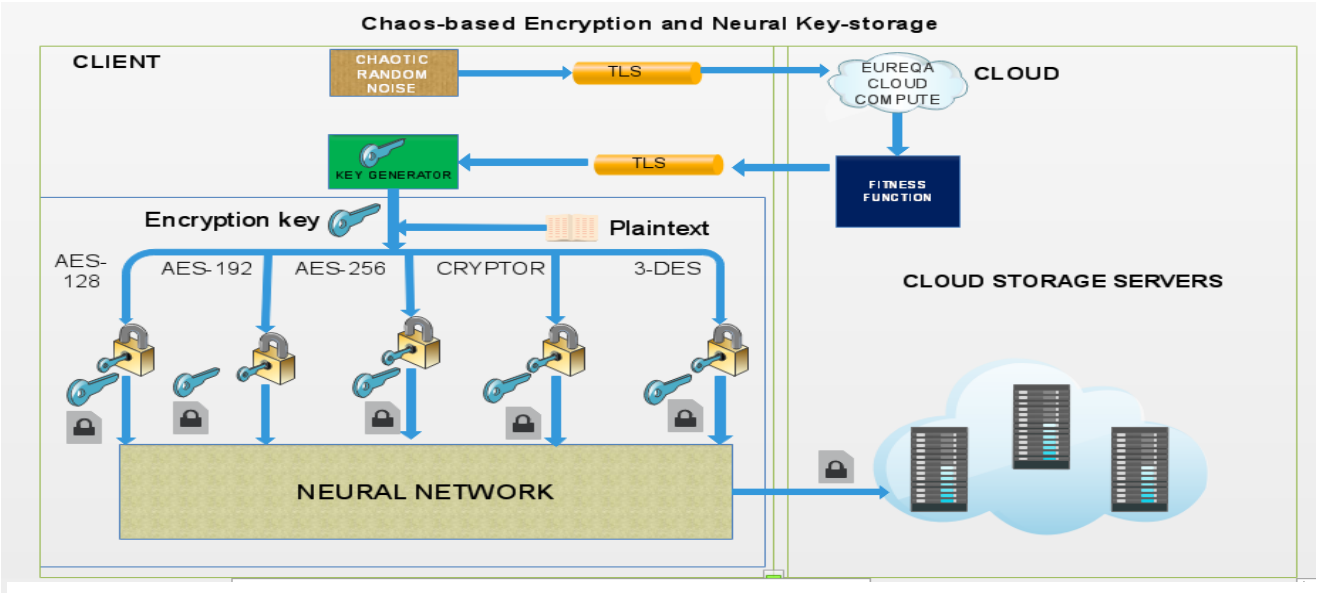


**Figure 1: Proposed model solution**

The model describes the process of encrypting cloud-bound data on the client-end and how encryption keys are-managed.

The process is initiated by generating random noise. The noise can be from any source. In this paper, the noise was sourced from random.org. The noise is then transmitted into the Eureqa cloud-based system through a secured channel using transport layer security (TLS). Eureqa implements an exhaustive search which mimics the concepts of evolutionary computing to generate a non-linear fitness function that fits the input noise. Hence, we argue that the fitness function is the best approximation of the input noise. The next section outlines the process of generating encryption keys from the resultant fitness function.

### *B. Encryption key generation process*

To generate random encryption keys, the fitness function is normalized using random floating point numbers in the range [0,1]. This produces a set of random outputs. The outputs are randomly picked and converted into a binary stream. The minimum length of the binary stream is 56-bits. If the output bit stream is less than 56 bits, the process is repeated and the random binary outputs are concatenated to form the desired length. The key length depends on the encryption algorithm to be used i.e. DES, 3-DES, AES and Cryptor. DES and Cryptor use a 56-bit encryption key. However, Cryptor can use resizable encryption keys. This means Cryptor uses keys of variable length, compared to DES which uses 56 bits only.

Hence, the strength of the encryption keys emanates from the random property of the floating-point numbers derived from the chaotic and random attributes of the input noise. It is on this premise, that the proposed system is believed to achieve strong encryption keys which are not reliant on the key length but chaos and randomness. The following algorithm summarises the encryption process.

### *C. Key generation and encryption algorithm*

Algorithm 1: Key generation and encryption

1. Generate random noise
2. Input random noise into the Eureqa system
3. Obtain a fitness function
4. Generate random floating-point numbers between [0,1]
5. Normalize the fitness function with random floats
6. Convert random outputs to binary to generate random encryption keys
   6.1. Pick an encryption key at random
   6.2. Check length of key depending on the encryption algorithm to be used
7. Encrypt plaintext
8. Send ciphertext and key into the neural network
9. Neural network learns the key and ciphertext patterns
10. Neural network outputs ciphertext
11. Send ciphertxt to the cloud for storage
12. Discard the encryption key

The key generation phase in the algorithm above is generic to all the encryption schemes discussed herein. Step 7 in the algorithm is unique to each scheme. For example, Cryptor implements the encryption through an exclusive or i.e. *XOR* operation. Thus, the encryption is based on a one-time pad implementation. Therefore, the length of the key and the plaintext must be equal in order to encrypt data successfully. Steps 8-10 describe how the ciphertext and the encryption key are processed by the neural network. The ciphertext is used as input into a counter propagation neural network (CPNN) together with the encryption key. The CPNN is trained to learn patterns of the binary ciphertext and encryption key. Once the key has been learnt, it is then discarded to avoid getting compromised.

Decryption can only be done by the data owner. This is done in such a manner that the CSP or any unauthorized third party entities cannot decrypt the data. This provides the user with absolute assurance of the confidentiality guarantees of their cloud-hosted data. Once the ciphertext has been downloaded from the cloud, it is sent into neural network to be processed. The neural network uses unsupervised machine learning to reconstruct the key, from the ciphertext patterns. Fig.2 below presents the decryption model.

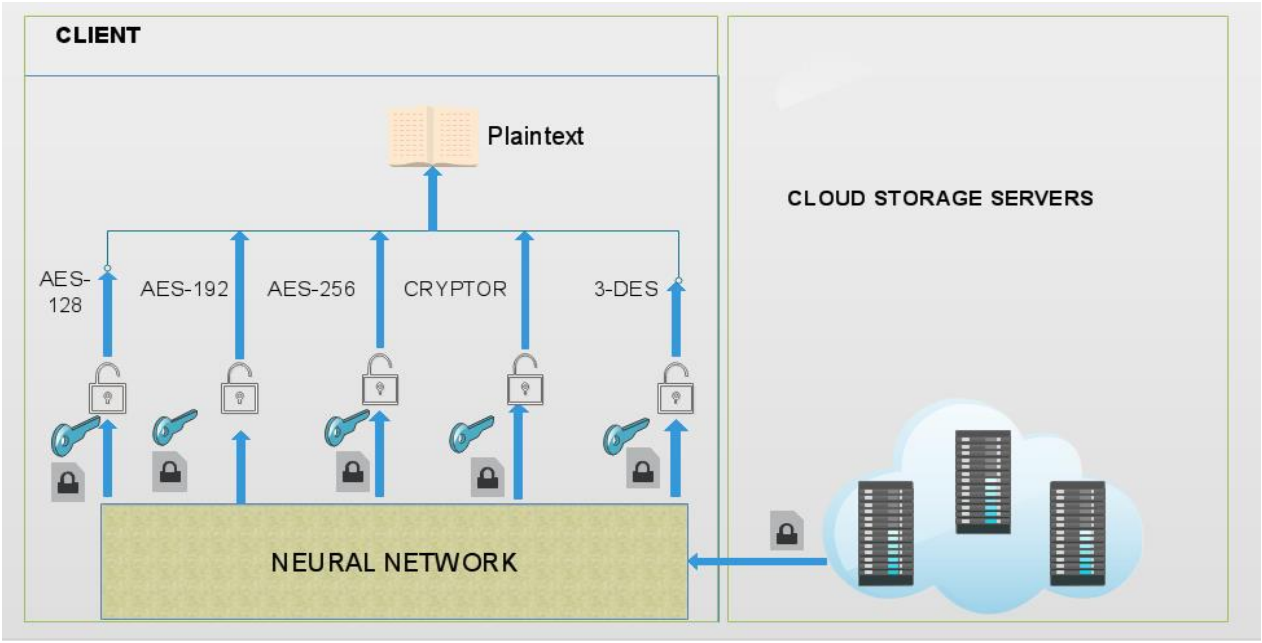


**Figure 2: Decryption model**

The output of the neural network is compared with the target value set during the training phase by computing the Euclidean distances. If the target value matches the ciphertext at each instance, the corresponding encryption key (which was used as input during encryption) is produced. The key value is converted back to binary form. The plaintext is recovered by performing the reverse of the encryption process. The encryption key is discarded once the decryption process halts. The next section discusses the results.

## IV. DISCUSSION OF RESULTS

The results presented herein compare Cryptor with DES, 3-DES, AES-128, AES-192 and AES-256. These cryptosystems were chosen because they are also symmetric and are widely used for data encryption. Hence, they have some similarities with Cryptor.

A cloud infrastructure was set-up in order to test the application and practicability of the proposed model solution on a live cloud infrastructure. We used OpenNebula 4.12.3 which comes with a kernel-based virtual machine (KVM) hypervisor. OpenNebula does not have an encryption module. Thus, the proposed system can be integrated into the cloud infrastructure easily to provide a client-end security service to cloud users.

The encryption schemes were used to encrypt a text file of 167 bytes in size. Fig.3 depicts a table with the overall performance of the encryption schemes when encrypting the text file. It also shows the CPU and memory use in percentages.

| Cyptosystem | Crypt | DES | 3-DES | AES 128 | AES 192 | AES 256 |
|---|---|---|---|---|---|---|
| Key Size (bit) | 56 ≥ | 64 | 192 | 128 | 192 | 256 |
| File Sizes (byte) | 167 | 167 | 167 | 167 | 167 | 167 |
| Encryption Time (ms) | 126.1 | 12.4 | 13.4 | 87.9 | 164.5 | 149.6 |
| Decryption Time (ms) | 128.3 | 10 | 14 | 67.7 | 143.7 | 130.4 |
| Memory (mb) | 8 | 4 | 4 | 6.3 | 11.1 | 15.3 |
| CPU (percent) | 1.008739 | 1.286582 | 1.321932 | 1.004848 | 1.006667 | 1.0094462 |

**Figure 3: Comparison of cryptosystems**

In terms of computing resources such as CPU and memory, Cryptor has better results compared to DES and 3-DES. This results mean that Cryptor is indeed a light-weight cryptosystem. It therefore meets the functional requirements. Thus, the cryptosystem can be deployed on computing devices with low memory and CPU specifications given that it is designed to be a client-end cryptosystem. Most client devices have low specifications. For example, mobile devices. Hence, Cryptor can be executed from them.

In terms of encryption and decryption times, Cryptor performed better than AES-192 and AES-256. These results show that the encryption scheme is indeed efficient and achieves its objective.

Fig. 4 depicts the encryption and decryption times, measured in milliseconds for all five encryption schemes.

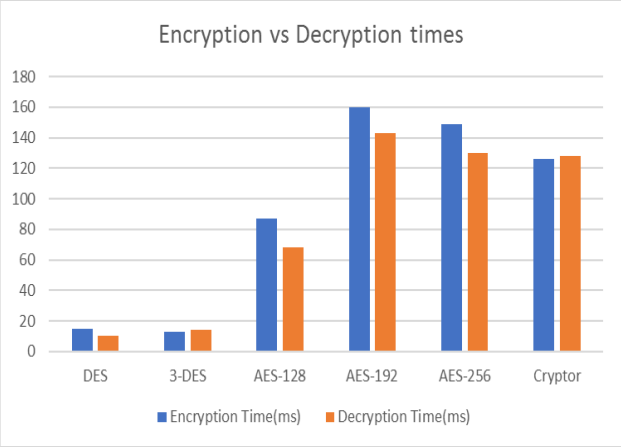


**Figure 4: Encryption vs Decryption times**

Cryptor took 126 milliseconds to encrypt the text file. DES encrypted the text file in 12 milliseconds. 3-DES encrypted the text file in 13 milliseconds. AES-128 took 87 milliseconds, AES-192 took 160 milliseconds and AES-256 recorded 149 milliseconds to encrypt the same text file. In this regard, only DES, 3-DES and AES-128 encrypted the text file faster than Cryptor. The former encrypts the text file with the least amount of time. Cryptor outperformed AES (192 and 256). These results were obtained after ten consecutive runs. On average, Cryptor performs comparatively better that the other two variants of AES (i.e. AES-192 and AES-256).

## V. Conclusion and Future Work

The proposed scheme uses chaotic random noise to improve the strength of encryption keys. The strength of the encryption keys does not rely on the length of the key but the random and chaotic nature of the input noise. The encryption keys can be used to improve the strength of existing cryptosystems such as DES, 3-DES, AES. Overall, it is concluded, based on the results, that Cryptor is a lightweight, strong client-end encryption scheme. The results make Cryptor a better encryption scheme in terms of encryption and decryption times. To enforce data confidentiality guarantees, various security issues still need to be analysed. As such, future perspectives of include: experimenting on encrypting multi-media digital content, implement the Cryptor system to have rounds of encryption.

## Acknowledgments

The support of the National University of Lesotho (NUL), University of KwaZulu-Natal (UKZN) and the University of Pretoria (UP) is acknowledged. Special thanks go to Lerato Lerato, Teboho Khoali and Kopano Moeketsi, for their intriguing ideas on crypto. Opinions and conclusions reached by this paper are those of the authors and should not be attributed to the NUL, UKZN or UP.

## References

[1] A. Shawish and M. Salama, 2014. Cloud Computing: Paradigms and Technologies, F. Xhafa and N. Bessis (eds.), Inter-cooperative Collective Intelligence: Techniques and Applications, Studies in Computational Intelligence 495, DOI: 10.1007/9783-642-35016-0_2, Springer-Verlag Berlin Heidelberg.

[2] "IT Security - Dropbox Business", *Dropbox.com*, 2016. [Online]. Available: https://www.dropbox.com/enterprise/security. [Accessed: 03- Mar- 2017].

[3] D. Newton, "Dropbox authentication: insecure by design", *Dereknewton.com*, 2011. [Online]. Available: http://dereknewton.com/2011/04/dropbox-authentication-static-host-ids/. [Accessed: 28- Feb- 2017].

[4] Mark D. Ryan. Cloud computing privacy concerns on our doorstep. Com-munications of the ACM, 54(1), 2011.

[5] T. Security, "2016 Global Encryption and Key Management Trends", Thales-esecurity.com, 2017. [Online]. Available: https://www.thales-esecurity.com/cpn/2016-global-encryption-trends-study. [Accessed: 14- Mar- 2017].

[6] S. Anthony, "Google teaches "AIs" to invent their own crypto and avoid eavesdropping", *Ars Technica*, 2016. [Online]. Available: https://arstechnica.com/information-technology/2016/10/google-ai-neural-network-cryptography/. [Accessed: 22- Mar- 2017].

[7] W. Kinzel, and I. Kanter, "Neural Cryptography", Proc. of the 9th Int'l Conf. on Neural Information Processing (ICONIP'02), vol. 3, pp.1351-1354, 2002.

[8] D. A. Karras, and V. Zorkadis, "On neural network techniques in the secure management of communication systems through improving and quality assessing pseudorandom stream generators", Neural Networks, vol.16, issues 5-6, pp. 899-905, June-July 2003.

[9] R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. In Foundations on Secure Computation, Academia Press, pages 170–173, 1978.

[10] R. Chirgwin, "Researchers crack homomorphic encryption", *Theregister.co.uk*, 2016. [Online]. Available: https://www.theregister.co.uk/2016/08/16/researchers_crack_homomorphic_encryption/. [Accessed: 22- Mar- 2017].

[11] D. Boneh and R. Lipton. Searching for Elements in Black-Box Fields and Applications. In Proc of Crypto '96, LNCS 1109, pages 283–297. Springer, 1996.

[12] W. van Dam, S. Hallgren, and L. Ip. Quantum algorithms for some hidden shift problems. In Proc. of SODA '03, pages 489–498. Full version in SIAM J. Comput. 36(3): 763–778 (2006).

[13] C. Gentry. A fully homomorphic encryption scheme. PhD thesis, Stanford, CA, USA, 2009

[14] C. Van Dijk, M., and Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT'10, pages 24–43, Berlin, Heidelberg, 2010. Springer-Verlag.

[15] SAICSIT

[16] Kumar N., Katta V., Mishra H., and Garg H. (2014). Detection of Data Leakage in Cloud Computing Environments. 6Th International Conference on Computational Intelligence and Communication Networks, IEEE Computer Society; 803 – 807.

[17] Ratsoma, M.S., Dlamini, M.T., Eloff, J.H.P. and Venter, H.S. (2015). A Conflict-Aware Placement of Client VMs in Public Clouds. In: 10th International Conference on Cyber Warfare and Security. Reading: Academic Conferences and Publishing International Limited, pp.501-503

[18] Zissis, Dimitrios, and Dimitrios Lekkas. "Addressing cloud computing security issues." Future Generation computer systems 28, no. 3 (2012): 583-592.

[19] Damiani, Ernesto, S. D. C. D. Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. "Balancing confidentiality and efficiency in untrusted relational DBMSs." In Proceedings of the 10th ACM conference on Computer and communications security, pp. 93-102. ACM, 2003.

[20] Mosola N.N., Dlamini M.T., Eloff J.H.P., Eloff M.M. (2016). *Evolutionary Neural Crypto-System for Cloud-bound Data*. Southern Africa Telecommunications Networks and Applications Conference (SATNAC), George, South Africa.

**Napo Mosola** just completed his MSc in Computer Science, at the UKZN. He obtained a BSc and BSc (Hons) in Computer Science from the NUL in 2012 and University of Johannesburg (UJ) in 2015, respectively. He is currently a member of the Mathematics and Computer Science (MACS) department at NUL. His research interests include cybersecurity, machine learning, the Internet of Things (IoT), cloud computing.
Please address all correspondence to: nn.mosola@nul.ls