KOMPUTASI: JURNAL ILMIAH ILMU KOMPUTER DAN MATEMATIKA
Vol.21 (1) (2024), 83-91, p-ISSN: 693-7554, e-ISSN:2654-3990

Chaos CSPRNG Design As a Key in Symmetric Cryptography Using Logarithmic Functions

Hizkia Nathanael[1], **Alz Danny Wowor**[2,*]

[1]Informatics Engineering Study Program , Faculty of Information Technology, Satya Wacana Christian University, Central Java, 50714, Indonesia
[2]Department of Informatics Engineering, Faculty of Information Technology, Satya Wacana Christian University, Salatiga, Central Java, 50714, Indonesia

Abstract

Abstract This research uses the logarithm function as a key component in generating random numbers in the Chaos CSPRNG framework. The main problem addressed here is the generation of keys for cryptography, recognizing the important role of cryptographic keys in safeguarding sensitive information. By using mathematical functions, speci cally logarithmic functions, as a key generation method, this research explores the potential for increasing the uncertainty and strength of cryptographic keys. The proposed approach involves the systematic utilization of various mathematical functions to generate diverse and unpredictable data sets. This data set, derived from the application of logarithmic functions, serves as the basis for generating random numbers. Through a series of tests such as Randomness Test and Cryptography Test, this research shows that the data generated from these functions can be utilized e ectively as a reliable source for generating random numbers, and has a low correlation value, thereby contributing to the overall security of a symmetric cryptographic system.

Keywords: *Chaos CSPRNG, Symmetric Cryptography, Logarithmic Functions*

1. Introduction

The need for information security has undergone a significant transformation in the eld of cryptography. The role of the key has become crucial and serves as a primary indicator in the design of cryptography or information security. Therefore, the design of every algorithm needs to consider the key generation process that is di cult to predict and can obscure plaintext and ciphertext. Hence, algorithms need to be designed to provide robust security. Symmetric cryptography has become a critical aspect of information security in the digital era.

Secure Pseudorandom Number Generators (CSPRNGs) are key elements in many security applications, including data encryption and authentication. CSPRNGs generate random numbers used as keys in various security protocols. Various methods are employed to create keys with good quality. The use of chaos in CSPRNG is a promising approach to generate strong random numbers. Chaos refers to the dynamic change phenomenon highly sensitive to initial conditions, appearing random and ensuring that the algorithm adheres to Shannon's principles.

[*]**Corresponding author:** ***E-mail adress:*** *alzdanny.wowor@uksw.edu*
Received: 25 Nov 2023, Accepted: Accepted: 26 Jan 2024 and available online 30 Jan 2024
DOI: https://doi.org/10.33751/komputasi.v21i1.9265

Many generator functions can produce CSPRNG-based random numbers, with the logistic function $f(x) = rx(1\bar{x})$ being the most famous, interatively given by $x_{i+1} = rx_i(1-x_i)$. In research [2] - [5], it is used as a complement in the algorithm. Research [6] tests polynomial functions of degree-1, degree-2, and degree-3, and then transforms them into iterative functions with xed-point iteration. This research successfully generates CSPRNG chaos-based keys. Research [7] tests the function $f(x) = x^2\_9x - 99$ an iterative function, resulting in CSPRNG chaos-based random numbers with correlation levels closest to zero. Research [8] regenerates trigonometric functions and uses them as CSPNRG chaos-based random number generators. Research [9] Implements the cubic function $f(x) = 3(x^3\_x^2\_x)$ using xed-point iteration to generate several iterative functions that can be used as random number generators. Research [10] regenerates constants and coe cients of linear functions based on CSPRNG chaos, used as a Flexible S-box Design.

One interesting method for generating random numbers is using logarithmic functions. These functions have complex properties and are challenging to invert, making them interesting candidates for use in cryptographic algorithms. The use of logarithmic functions to generate random numbers can create a series of numbers that are very di cult to predict. Current research employs logarithmic functions as random number generators. Visualization testing using scatter plots, randomness testing, and encryption processes are the testing methods to ensure that the numbers generated by logarithmic functions can be used as random number generators.

2. Methods

The stages in the research process, as shown in Figure 1, involve initializing x0 for logarithmic functions. In the iteration process, 200 iterations are taken and divided into 5 datasets, with each dataset containing 3 numbers in sequential order. Testing is conducted for each dataset to determine whether the numbers are random or non-random, proceeding to the graphical test process.

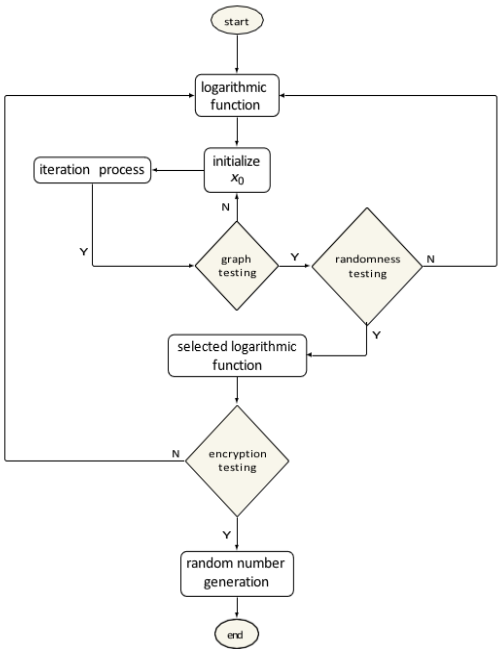

Figure 1. Research Process Scheme

In the graphical test process, if the visualization appears non-random, the process will return to the initialization x0. However, if it successfully shows a random visualization, the random testing phase will continue. This random test includes three types of tests: the Run test, Mono Bit test, and Block Bit test. If successful, the selected logarithmic function will proceed to the encryption

test. In the encryption test process, correlation values are used as a reference. If the correlation value approaches zero, the function can be considered one of the good random number generators.

3. Result and Discussion

3.1. Selected Functions

Some functions that will be used for random number generation, as shown in Table 1, are listed below.

Table 1. Fungsi Pembangkit Terpilih

No	Generator Function
1	$f(x) = logx^{2.647}$
2	$f(x) = logx$
3	$f(x) = log2.647\sqrt{x^2}$
4	$f(x) = log2.647\sqrt{x}$
5	$f(x) = lnx\sqrt{x^2+1}$
6	$f(x) = ln\sqrt{x}$
7	$f(x) = lnx$

The functions listed in Table 1 will be processed to generate random numbers. If a function produces non-sequential numbers when generated for a total of n times, and the function does not encounter errors, it will proceed to the next stages. If it passes the randomness test, then the function can be considered as a random number generator.

The rst test uses the function $f(x) = logx^{2.647}$ with $x0 = 10.57$. Based on Equation 1, an iterative function is obtained. The results of the rst ten iterations are provided in Table 2.

Table 2. Results of Iteration for the Function $f(x) = logx^{2.647}$

i	x	Data 1	Data 2	Data 3	Data 4	Data 5
1	2.710726491402760	710	726	491	402	76
2	1.146377848089250	146	377	848	089	25
3	0.157040648025306	157	040	648	025	306
4	2.128156027671230	128	156	027	671	23
5	0.868225173126667	868	225	173	126	667
6	0.162440107286225	162	440	107	286	225
7	2.089294920517260	089	294	920	517	26
8	0.847039334045419	847	039	334	045	419
9	0.190839228576645	190	839	228	576	645
10	1.904072724214620	904	072	724	214	62

Table 2 shows how to extract integers from the mantissa in the function, generating 5 data points. This study takes three digits for each data point, considering the maximum number of digits from the ASCII characters, which is 256 characters.

The iteration results are visualized using scatter plots in Figure 2 for the rst 200 iterations. Then, all generated functions will be visualized with Scatter Plots for n = 200, resulting in:

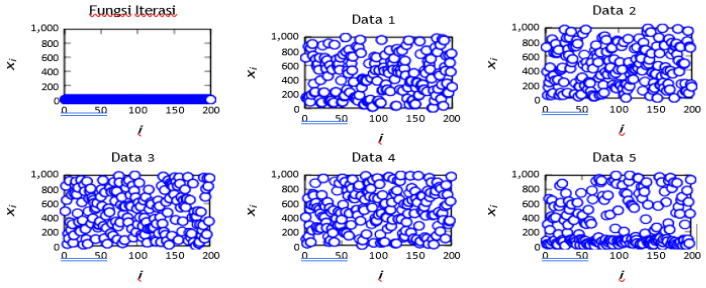

Figure 2. Results of Iteration for the Function $f(x) = logx^{2.647}$

As seen in Figure 2, the values generated by the function $f(x) = logx^{2.647}$ visually show that all the data points on the Scatter Plots diagram appear to form chaos. The ve data points can be further examined in the next stage of testing.

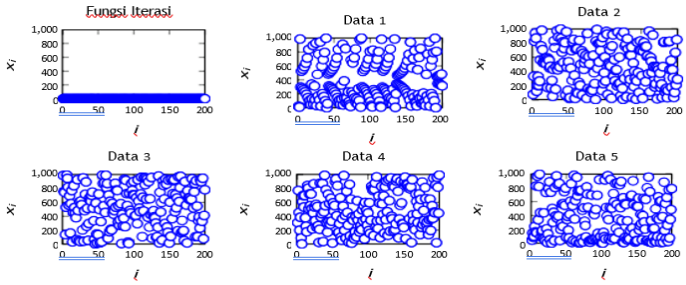

Figure 3. Results of Iteration for the Function $f(x) = logx$

As seen in Figure 3, the values generated by the function $f(x) = logx$ visually indicate that only data-2, data-3, data-4, and data-5 in the scatter plot diagram seem to exhibit chaos. The four data points can be further examined in the next testing stage, but data-1 does not pass the visual inspection. Despite the diagram generated by data-1 appearing random, it still forms a pattern.

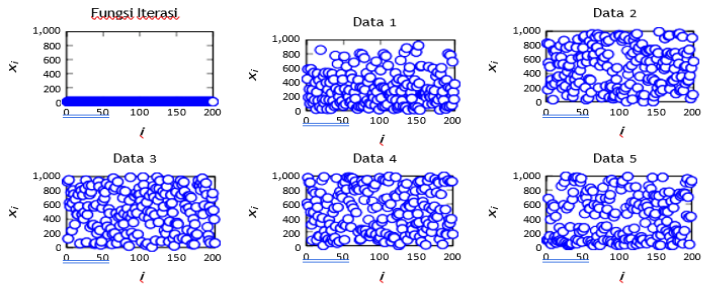

Figure 4. Results of Iteration for the Function $f(x) = log2.647\sqrt{x}$

As observed in Figure 4, the values generated by the function $f(x) = log2.647\sqrt{x}$ visually demonstrate that all the data points in the Scatter Plots diagram appear to form chaos. The ve data points can be further examined in the next stage of testing.

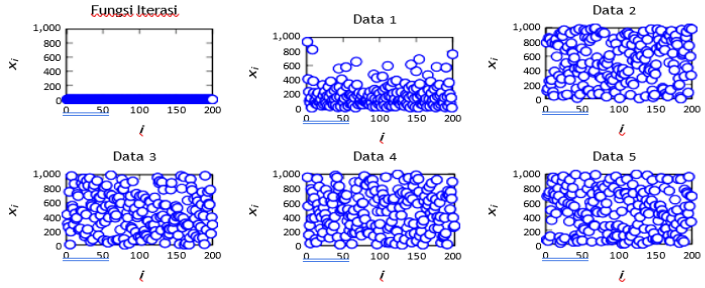

Figure 5. Results of Iteration for the Function $f(x) = log2.647\sqrt{x}$

As seen in Figure 5, the values generated by the function $f(x) = log2.647\sqrt{x}$ visually show that all the data points in the Scatter Plots diagram appear to form chaos. The ve data points can be further examined in the next stage of testing.

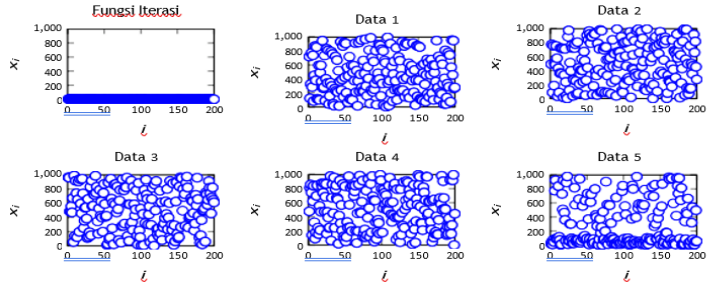

Figure 6. Results of Iteration for the Function $f(x) = lnx\sqrt{x^2+1}$

As observed in Figure 6, the values generated by the function $f(x) = lnx\sqrt{x^2+1}$ visually indicate that only data-1, data-2, data-3, and data-4 in the Scatter Plot diagram seem to exhibit chaos. The four data points can be further examined in the next testing stage, but data-5 does not pass the visual inspection. Despite the diagram generated by data-5 appearing random, it still forms a pattern.

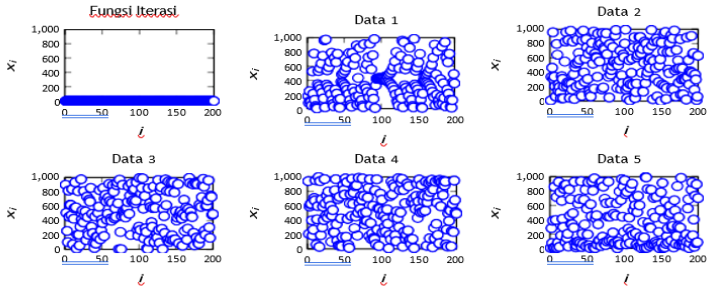

Figure 7. Results of Iteration for the Function $f(x) = ln\sqrt{x}$

As seen in Figure 7, the values generated by the function $f(x) = ln\sqrt{x}$ visually indicate that only data-2, data-3, data-4, and data-5 in the Scatter Plots diagram seem to exhibit chaos. The four data points can be further examined in the next testing stage, but data-1 does not pass the visual inspection. Despite the diagram generated by data-1 appearing random, it still forms a pattern.

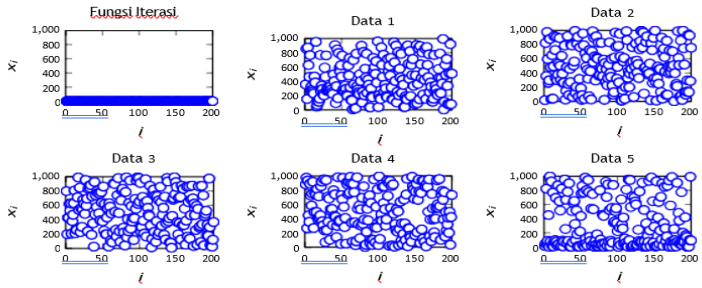

Figure 8. Results of Iteration for the Function $f(x) = lnx$

As seen in Figure 8, the values generated by the function $f(x) = lnx$ visually show that only data-1, data-2, data-3, and data-4 in the Scatter Plots diagram seem to exhibit chaos. The four data points can be further examined in the next testing stage, but data-5 does not pass the visual inspection. Despite the diagram generated by data-5 appearing random, it still forms a pattern.

3.2. Randomness Test

After being visualized with Scatter Plots generated by each function, the next step is testing random numbers using the Run Test and Mono bit methods to determine the randomness of a dataset. The testing is conducted with a significance level of $\alpha = 1\%$. The criteria for numbers to be considered random are if the p-value is $> \alpha$, and vice versa if the p-value is $< \alpha$. With $x_0 = 10.57$, the results are as follows:

Table 3. Results of Randomness Testing on the Seven Functions

$f(x)$	Data	Run Test	Mono Bit	Result
$log(x^{2.647})$	Data 2	0.26383	0.16563	random
$log(x^{2.647})$	Data 3	0.84745	0.10740	random
$log(x^{2.647})$	Data 4	0.24492	0.37857	random
$log(x)$	Data 2	0.96908	0.01226	random
$log(x)$	Data 3	0.47975	0.01079	random
$log(x)$	Data 4	0.01079	0.47975	random
$log(2.647\sqrt{x^2})$	Data 2	0.79121	0.68732	random
$log(2.647\sqrt{x^2})$	Data 3	0.22638	0.75424	random
$log(2.647\sqrt{x^2})$	Data 4	0.75318	0.15240	random
$log(2.647\sqrt{x})$	Data 2	0.35069	0.12837	random
$log(2.647\sqrt{x})$	Data 4	0.15486	0.21049	random
$log(2.647\sqrt{x})$	Data 5	0.11335	0.54511	random
$ln(x\sqrt{x^2+1}$	Data 1	0.34508	0.13999	random
$ln(x\sqrt{x^2+1}$	Data 2	0.21411	0.02004	random
$ln(x\sqrt{x^2+1}$	Data 3	0.85438	0.17971	random
$ln(x\sqrt{x^2+1}$	Data 4	0.15410	0.19465	random
$ln(\sqrt{x})$	Data 2	0.25591	0.97987	random
$ln(\sqrt{x})$	Data 3	0.25591	0.97987	random
$ln(\sqrt{x})$	Data 4	0.93898	0.44709	random
$ln(x)$	Data 1	0.64102	0.02842	random
$ln(x)$	Data 2	0.50258	0.15240	random
$ln(x)$	Data 3	0.17850	0.68732	random
$ln(x)$	Data 4	0.53113	0.92873	random

In Table 3, the results of the Run Test and Mono Bit testing for iterative functions are presented. Among the seven functions, only a few data points obtained random results in the Run Test and Mono Bit tests, making them potential candidates for e cient key generation in securing information.

3.3. Cryptography Test

This test is conducted in two parts. The rst part involves the correlation test between plaintext and ciphertext. The second part is the butter y e ect test, where we observe whether small changes in the initialization x_0 result in signi cant changes in the ciphertext. This test aims to assess how well the key plays a role in obscuring the ciphertext.

Each iteration function that produces random numbers is used as a key. The test is conducted through the encryption process.

$$Ek : P + K = C \ (mod\ 256)$$

The ciphertext FTI UKSW Salatiga. is taken, and a key block size of 256 bits or equivalent to 32 characters is used. Testing was performed based on the research in [6], which has become a random number based on CSPRNG chaos to be used as a key. Thus, the key is 32 characters from each data, and encryption operations are performed based on the seven selected functions, testing the correlation between plaintext and ciphertext. The results of each test are provided in Table 4.

Correlation testing can be used to assess the strength of the key in the encryption process, determining how well the key can secure information in a block cipher. Negative or positive correlation values are not of great concern; what matters is how close the value is to 0. The results for all tests in Table 4 are very good, with correlation values very close to 0, whether inversely or directly proportional.

Table 4. Correlation Value Relationship Level

$f(x)$	Data Retrieval	Correlation Result	Level of Relationship
$log(x^{2.647})$	Data 2	-0.0595	Very Low
$log(x^{2.647})$	Data 3	0.1299	Very Low
$log(x^{2.647})$	Data 4	-0.1407	Very Low
$log(x)$	Data 2	-0.2719	Very Low
$log(x)$	Data 3	-0.0638	Very Low
$log(x)$	Data 4	0.4091	Moderate
$log(2.647\sqrt{x^2})$	Data 2	-0.0214	Very Low
$log(2.647\sqrt{x^2})$	Data 3	-0.0460	Very Low
$log(2.647\sqrt{x^2})$	Data 4	-0.0448	Very Low
$log(2.647\sqrt{x})$	Data 2	-0.0518	Very Low
$log(2.647\sqrt{x})$	Data 4	0.0255	Very Low
$log(2.647\sqrt{x})$	Data 5	0.1568	Very Low
$ln(x\sqrt{x^2+1}$	Data 1	-0.0059	Very Low
$ln(x\sqrt{x^2+1}$	Data 2	-0.0069	Very Low
$ln(x\sqrt{x^2+1}$	Data 3	0.1030	Very Low
$ln(x\sqrt{x^2+1}$	Data 4	-0.0344	Very Low
$ln(\sqrt{x})$	Data 2	0.2387	Low
$ln(\sqrt{x})$	Data 3	0.3021	Low
$ln(\sqrt{x})$	Data 4	0.0920	Very Low
$ln(x)$	Data 1	-0.0768	Very Low
$ln(x)$	Data 2	0.1206	Very Low
$ln(x)$	Data 3	-0.3287	Very Low
$ln(x)$	Data 4	-0.2521	Very Low

Table 4 shows the results of the correlation test for each function with the value $x_0 = 10.57$. Based on the results in the table, some functions obtained very low correlation values. In the function $f(x) = log(x)$, the fourth data obtained a moderate correlation value. Meanwhile, in the function $f(x) = ln\sqrt{x}$, the second and third data obtained low correlation values.

The test results indicate that functions have very low correlations, making these data suitable for use as keys due to their random properties, making it challenging to break the key. On the

other hand, other data still exhibit bias and patterns, resulting in high correlation values, making them easier to decrypt.

Figure 9 illustrates the butterfly effect test, using the plaintext FTI UKSW Salatiga Selamanya! with a block size of 256 bits or equivalent to 32 characters. The simulation was performed by selecting the function $f(x) = logx$, specifically for the fourth data, to be the key. The initialization values were set to $x_0 = 0.054321$ and $x_0 = 0.054322$, both constants with a very small difference of 10^{-6}.

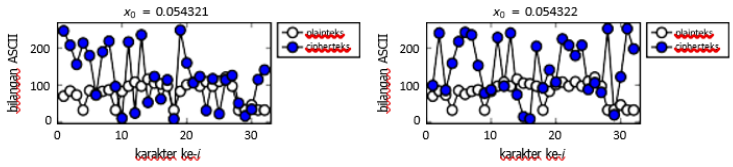

Figure 9. Comparison of Plaintexts and Ciphertexts FTI UKSW Salatiga Selamanya !.

In Figure 10, a retest was conducted using a different plaintext and combining numbers and symbols: 'ft1 uk$w $@L@t19@ $3l@m@ny@ 1, The tests were performed with the same block size and initialization value, resulting in a noticeable difference between the two.

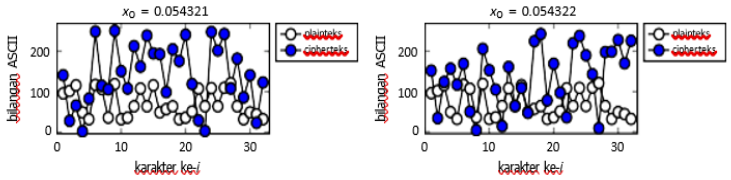

Figure 10. Comparison of Plaintexts and Ciphertexts 'ft1 uk$w $@L@t19@ $3l@m@ny@ 1,

In Figure 11, a retest was conducted using the same plaintext, with only one character being different: rrrrrrrsrrrrrrrrrrrrrrrrrrrrrrrrr. Both tests were performed with the same block size and initialization value, resulting in a noticeable difference between the two.

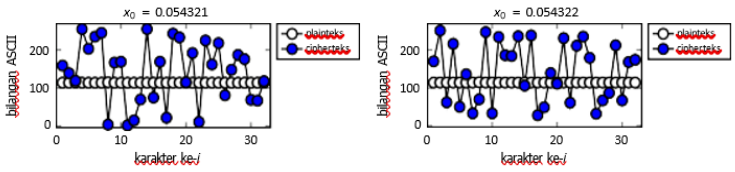

Figure 11. Comparison of Plaintexts and Ciphertexts rrrrrrrsrrrrrrrrrrrrrrrrrrrrrrrrr

The results of the tests are presented in Cartesian coordinates for plaintext and ciphertext, as shown in Figures 9 − 11. A small difference in the initialization value, on the order of 10 - 6 for the function $f(x) = log(x)$, can result in significantly different ciphertexts. This outcome indicates that logarithmic functions can exhibit the butterfly effect, where small changes in the input of the generator function can lead to substantial changes in the generated ciphertext.

4. Conclusion

Based on the research results, the number generator in regular logarithmic and natural logarithmic functions appears to produce both random and non-random numbers in the main iterations,

which can be observed visually. Some functions seem to form a pattern; therefore, the numbers are divided into several parts, namely data 1, data 2, data 3, data 4, and data 5. This division is done because there is a possibility that some data can generate random numbers. The Run Test, Mono Bit Test, and Block Bit Test indicate that some data successfully generate random numbers.

Correlation tests conducted on the functions $f(x) = log(x)$, especially data-4, and $f(x) = ln\sqrt{x}$, especially data-2 and data-3, resulted in an average correlation close to zero, indicating a low correlation category. This suggests that the use of these functions as regenerators is statistically unrelated in generating plaintext and ciphertext.

5. Acknowledgement

I would like to express my deepest gratitude to my parents for their unwavering support throughout my academic journey. Their love, encouragement, and sacrifices have been the driving force behind my success. I am truly grateful for everything they have done for me.

References

[1] A. Biryukov, J. Daemen, S. Lucks, and S. Vaudenay, Topics and Research Directions for Symmetric Cryptography, Early Symmetric Crypto workshop, 2017. [Online]. Available: https://orbilu.uni.lu/handle/10993/30953

[2] Sol s-S nchez, H., and Barrantes E.G., Using the Logistic Coupled Map for Public Key Cryptography under a Distributed Dynamics Encryption Scheme, Information, vol. 9, no. 7, pp. 1 12, 2018.

[3] Wowor, A.D. and V. B. Liwandouw, Domain Examination of Chaos Logistich Function As A Key Operator in Cryptography, International Journal of Electrical and Computer Engineering, Vol. 8, No. 6, pp. 4577-4583, 2018.

[4] Lawnik, M., Generalized Logistic Map and its Application in Chaos Based Cryptography, J. Phys. Conf, Ser., Vol. 936, No. 1, 2017.

[5] Ye, G., Jiao, K., Pan C., and Huang X., An E ective Framework for Chaotic Image Encryption Based on 3D Logistic Map, Hindawi Security and Communication Networks, Vol. 18, pp. 1-11, 2018.

[6] Wowor. A.D , Regenerasi Fungsi Polinomial Dalam Rancangan Algoritma Berbasis CSPNRG Chaos Sebagai Pembangkit Kunci Pada Kriptogra Block Cipher, Limits: Journal Of Mathematics And Its Applications,Vol. 14: pp. 1-15, 2017.

[7] Lihananto, D, "Regenerasi Fungsi dalam Pembangkit Bilangan Acak Berbasis CSRPNG Chaos," AITI, Vol. 16, No. 2, pp. 125 134, 2020.

[8] Paraditasari, K. and Wowor, A.D, "Desain Pembangkit Kunci Block Cipher Berbasis Csprng Chaos Menggunakan Fungsi Trigonometri, Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika, Vol. 6 : pp 400-405, 2021.

[9] Yopeng, M. R, and Wowor, A.D, The Implementation of f (x) = 3(x3 x2 x)+2 as CSPRNG Chaos-Based Random Number Generator, Indonesian Journal on Computing, Vol. 6, No. April, pp. 41 52, 2021.

[10] B. Susanto, A. D. Wowor, and V. B. Liwandouw, Desain S-Box Fleksibel: Regenerasi Konstanta dan Koe sien Fungsi Linier Berbasis CSPRNG Chaos, Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), Vol. 8, No. 1, pp. 7, 2019.