Security Improvement of the 256-BIT AES Algorithm With Dynamic S-Box Based on Static Parameter as the Key for S-Box Formation

Hendi Susanto [1,*], Alamsyah [1], Anggyi Trisnawan Putra [1]

[1] Department of Computer Science, Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang, Semarang, Indonesia
*Corresponding author: hendi.susanto@students.unnes.ac.id

ARTICLE INFO

Article history
Received: 5 Maret 2022
Revised: 15 Maret 2022
Accepted: 12 April 2022

Keywords
AES
S-BOX
Dynamic S-BOX
Pseudo Random Number Generator

ABSTRACT

Strong S-BOX is required for the usage of AES encryption and decryption process. Despite using a strong S-BOX that has better security, the usage of AES static S-BOX is something we can improve on. Symmetric encryption algorithm that uses block cipher is better if combined with make the S-BOX dynamic, but the problems occurred when we put totally random S-BOX into it because S-BOX has its own secure measurement. S-BOX that used in AES must have been tested having strong nonlinearity. In this research, we take collections of strongly tested S-BOX to be used on AES encryption and decryption processes. Because S-BOX that we used for encryption and decryption must be the same S-BOX, we added static parameters on both encryption and decryption process for the key in choosing which S-BOX we are used with pseudo random number generator (Pseudo RNG) for the algorithm. In practice, text input, hardware id's, or other variables can be used as the static parameter we used on this process. Pseudo RNG will take the numbers of S-BOX-es we had for variable of maximum output with minimum is 1, so we can always get the index of the S-BOX chosen. The purpose of this research is to add complexity to both the algorithm and security with dynamizing the S-BOX so we have more possibility of output which makes it harder to be attacked. The test result in this research is also tested with SAC test showing the average of 0,499 which is better than the regular AES with 0,504 and the nonlinearity is the same as regular AES which is 112.

1 Introduction

Advanced Encryption Standard (AES) has 3 types of standard key lengths, namely 128, 192 and 256-bit keys. The longer the key used, the more secure the resulting encryption (Munir, 2004). AES is included in the Block Cipher category, which is a cryptographic algorithm scheme that divides the data to be encrypted into blocks of the same length, in this case AES sets 3 block lengths, which are referred to as AES-128, AES-192 and AES-256 where the number behind it represents the block length in bits (Juremi et al., 2017). All Block Cipher algorithm systems utilize Substitution Box or S-BOX in their operations, unlike the Data Encryption Standard (DES) algorithm which uses a different S-BOX in each operation, AES uses a predetermined static S-BOX (Alamsyah et al., 2018).

Therefore, if someone either intentionally or unintentionally gets the ciphertext and the key used for encryption, that person can find out the contents of the message or information easily by decrypting the ciphertext with the usual AES algorithm (Manjula & Mohan, 2016). Although AES determines to use one fixed S-BOX, the AES algorithm can still run using different S-BOX for

encryption and decryption, provided the matrix size must be the same as the AES S-BOX (Al-Wattar et al., 2015). If at the time of encryption using a different S-BOX, the results of the calculation or encryption also produce a different value, so that if a data that has been encrypted with a different S-BOX and the key is known by an irresponsible person, that person cannot know the contents of the encrypted data, because in the decryption process it uses a different S-BOX from the encryption process of the data (Ibrahim, 2017).

However, not all S-BOX can be used freely, S-BOX has its own level of security, as explained by AES inventors, that S-BOX in AES must meet the nonlinearity criteria (Alsalami et al., 2016; Hallappanavar et al., 2014), because S-BOX is the only nonlinear element in encryption and decryption AES. The nonlinearity element in the S-BOX can be fulfilled if an S-BOX has a nonlinearity value of 112 or more (Daemen & Rijmen, 1998). Although encryption has been declared safe from nonlinear attacks, it cannot guarantee that the encryption is 100% safe, because if we use the same S-BOX for all encryption, and one day there is fraud where one party finds the encryption file and has the key used, the party can easily decrypt it with native AES algorithm with fixed S-BOX and get all the data stored in it. It's different if we use a different S-BOX for each encryption, even though one party already knows the key used, that party can not open the contents of the file easily, because the encryption process uses a different S-BOX, in other words S-BOX. S-BOX can be used as a second encryption key in the AES algorithm.

Therefore, a new problem arises, how to ensure that the S-BOX used during the encryption process is the same S-BOX used during the decryption process. For this reason, fixed parameters are needed for both processes (encryption and decryption) and not only limited to keys because they will be vulnerable to being known by these parties. This can be overcome by using the encryption key and static parameters in the program, these static parameters do not only have to be written to the program but can also use the unique identification of the device used (eg. mac address, device id, or others). Therefore, this study aims to modify the S-BOX used in encryption and decryption to be dynamic, of course by considering the security element in the S-BOX.

2 Method

2.1 Related Research

This research was developed from several references that are related to the research method and object. The use of this reference is intended to provide limitations and applications that will later be developed further. Research conducted by Vijay (Hallappanavar et al., 2014) entitled “Efficient Implementation of AES By Modifying S-BOX” modifies the S-BOX by changing the affine transformation formula, i.e. changing the equation.

$$X8 + X4 + X3 + X + 1 \text{ into } X8 + X6 + X5 + X + 1, \quad (1)$$

Resulting in a different S-BOX. This is applied with the aim of getting an algorithm that is lightweight or can be run with a minimum of resources to encrypt multimedia data, and the results of their research conclude that modified AES has better results than standard AES in Hamming Distance, stable output (Balanced Output) and Avalanche Effect.

Research by Arrag (Arrag et al., 2013) entitled “Implementation of Stronger AES by Using Dynamic S-BOX Dependent of Master Key” proposes a more complex and secure AES development method, namely by changing the S-BOX by performing XOR operations on each column in the S-BOX with the first byte of the encryption key. The results of this study are increasing complexity, increasing the distribution of results and randomization of encryption results, so that the security of the AES algorithm increases.

Research by Jacob (Jacob et al., 2015) entitled “Towards the Generation of a Dynamic Key-Dependent S-BOX to Enchance Security” also studying about converting the original S-BOX from AES by using a function called Codeword Generator which is based on each key entered and generates a new S-BOX for the AES encryption and decryption process. The results of the application of dynamic S-BOX in this study proved to be safer than AES with the S-BOX standard.

Research by Alamsyah (Alamsyah et al., 2018) entitled “The replacement of irreducible polynomial and affine mapping for the construction of a strong S-box” modifying S-BOX using

irreducible polynomial method with 8-bit constant vector and AES Affine Mapping. The results of the S-BOX were applied with AES and concluded that the proposed S-BOX is safer than AES with the regular S-BOX, this is evidenced by the tests carried out by testing balance, bijective, nonlinearity, SAC, and BIC (BIC-nonlinearity).

2.2 Research Approach

This research uses pseudo random number generators to dynamically choose the S-BOX we'll be used on the encryption and decryption process of AES 256-bit. It is expected to be able to improve security of the encryption result or ciphertext. The research algorithm used in this study can be seen in Figure 1.

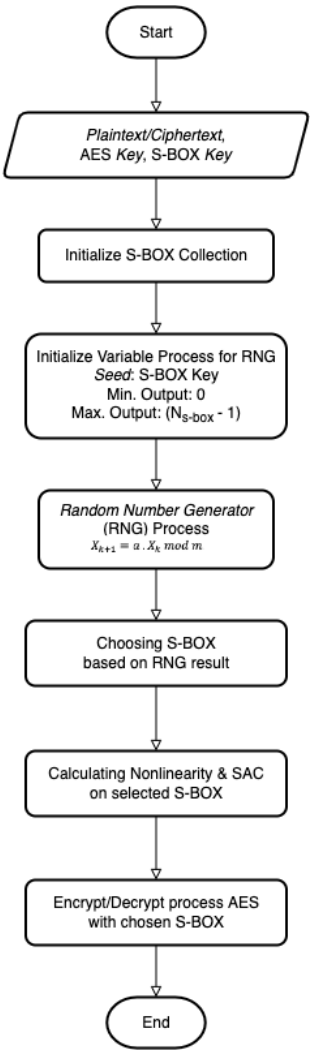

Figure 1. Research Algorithm

As shown on Figure 1, the main method proposed in this research is on the usage of collection S-BOX-es and randomly choose one of the S-BOX to use on encryption or decryption process. To make sure we use the same S-BOX on both encryption and decryption, we used static variable or in this case we called S-BOX Key that we passed on to Pseudo Random Number Generator to get the index of S-BOX we will use. After we get the S-BOX, the overall process of encryption and decryption is the same as the standard of AES.

2.3 Experiment Data

This study uses text as an experimental object for the proposed method. In this study, 25 experiments will be carried out, both from the standard AES method and the AES method with dynamic S-BOX. The data used in plaintext refers to the "Lorem Ipsum" text which is taken sequentially with a character length of 32 characters for each text or data that represents 256 bits.

While the primary key data is taken from random alphanumeric characters with the same character length, which is 32 characters, and the last one is “S-BOX Key” with varying length by dividing 5 test data for each “S-BOX Key” length. Text data shown on table 1.

Table 1. Plaintext used on algorithm testing

#	Plaintext	Encryption Key	S-BOX Key
1	Lorem ipsum dolor sit amet, cons	Jb0Vt0fKRw8t06VgRIjef5oBWjmufiTW	Nemju6PtHufFUAIf0c3apKgLWurWCclv
2	nsectetur adipiscing elit. Etiam	nP0Uxkb98y5yjglcvJMDjIlIVJAXfNlz	u9omgEPlPU9qLhrevAolVqoS7vBoXh7d
3	dignissim, erat at dictum alique	oip7MQmPK5kp3U5fjH4vKTVhKUDPR7Mj	6sPzElrxWYr4HkiVflRkDO6uPUgQb2zD
4	ligula nulla malesuada magna, in	ivEpcJ6OoU1dqMf8MGUTXZIb5uSSlLum	0NsX2BJu5W3YDuqlgB9nd8Yq9ql0ckO3
5	aliquet massa tellus id velit. N	HzzkmP4y8VHbNWfDDpn2fQqkxgguwYOT	8DVwa2JQiGjLhTgKQvjYVZLNr3L2Aqwl
6	ullamcorper diam eget massa posu	wiby3OTqBqikIrexbQO6wVOvNeXEsGNF	ZY2Hbea1ixZQsSBeqKeEdZNa
7	vitae condimentum ante pulvinar.	VK4u30Ft6gejjXidf46fBDo9h0vX7HIi	6Jdq6mVL1JvesXvcHj5vRd4t
8	Nulla bibendum molestie odio. Du	7Ir58ttO525IVHJ3s0bGIPZnfgcyI6Zq	At8lcvI9JXVI2Eoi2FBtnUja
9	Duis quis lorem accumsan, egesta	YvXdAC6qPzXWgwhhlg53hJamKQ1wsgMk	6VHT8iApmp2Ik3eX9m9W0bML
10	felis sit amet, vestibulum eros.	XBtf37BwKRyZ1IQMxwQ3Y2YMyAYhoKKL	1ewGBIscG0mG46ntwtlywoMt
11	dictum massa et nisl suscipit ul	JlYEr2DxD5SdLhnNFPmDaG8M6emBe5tK	sjU8b9FkSVkKVqn1
12	Sed ultricies sed purus eget ves	1LYqoaGA1KClPLDvlNeuzMJyobjedbO1	eHOqcnnJ267AvdpD
13	.Nulla facilisi. Donec vulputate	5fCLWOXio4Pf6GJcYPyPria83JmDNCvk	c7XtfAncUn4UVnKE
14	sapien ac vestibulum. Suspendiss	R6pICCZXmMnNzVIv73deSZ8Rk50zdalH	ZL20xGRl9GWys6rs
15	suscipit nibh ex, a blandit nibh	s6klEivVu7PkahdchqxdVGzF3swNWFdf	mA8DJJxNsc4LAANc
16	maximus sit amet. Sed pulvinar m	14MzimCbbxrYGmGm7KUslmvS86agFZZn	PHcSFHfa
17	Sed placerat mauris sit amet sag	KzkVh9Pm7hm64YXlFpumP6bTj7XlovPP	bhASHCgk
18	gravida. Vivamus sit amet finibu	dZIVFu4vEoIlbzuLYSzaXKQA0M6fSu5k	EnX1OMI9
19	tellus, ac iaculis turpis. Vesti	zSABi4AphuVOzM1BWb65FpOut67fc8tu	A5vcOW2e
20	Vestibulum ante ipsum primis in.	LokIgpONCWO76pzpEDZHk16wPmTjpd36	tEKpRhCM
21	faucibus orci luctus et ultrices	QVM5pygEFV6KOQZIwVJJ6fptViNl3UtH	yEPS
22	cubilia curae; Lorem ipsum dolor	2JUzAGrag7v6ddFFX6bGQw8vvfychOxM	GPWb
23	sit amet, consectetur adipiscing	sW1VkbSgiapqXY9sydIBLeuftWwoEvW8	n1wR
24	elit. Interdum et malesuada fame	SguhpEppOnafJwvtSr6CCSRXb4oxy8Qn	c31w
25	ac ante ipsum primis in faucibus	fCKVJniGSC2uIcnZCv2bjJuLjWUijHZN	8wnm

Despite data shown on table 1, on this study, we also used S-BOX-es from Alamsyah (Alamsyah et al., 2018) research results. This S-BOX-es will be used on every testing we try.

2.4 Experimental Stages

2.4.1 *Algorithm Implementation*

The algorithm is built using the Java programming language based on a Graphical User Interface (GUI) with a website platform. The development of the algorithm is compiled using the help of the Intellij IDEA application. The implementation of the Java programming language into the website platform is built using the Spring framework, in the development of dynamic S-BOX algorithms, assisted by the Random library contained in the java.util.random library collection to generate random numbers in dynamic S-BOX selection, other than the library , the algorithm is built without other help libraries, such as Cipher which contains the AES algorithm in it.

2.4.2 *Algorithm Constructions*

Basically, the dynamic S-BOX selection process that will be used in the encryption and decryption process uses the Random Number Generator (RNG) algorithm to determine the S-BOX index used. Random Number Generator was chosen as the selection algorithm because it can generate random numbers but is consistent with the given parameters.

The Random Number Generator Algorithm used in this study uses the help of the Random library developed by the Java utility library in the java.util.Random package, this library was chosen due to the flexibility of the parameters used to form random numbers that can be changed as needed. This library uses a linear congruential pseudorandom number generator algorithm defined by D. H. Lehmer with the following formula.

$$Xk+1=a \,.\, Xk \bmod m \tag{2}$$

In the Random Number Generator, the parameters used to generate random numbers include the seed, the minimum output number, and the maximum output number. The parameters included in this algorithm are S-BOX Key for X0 (seed), first index or zero (0) as the minimum output, and ns-box-1 for maximum output. After the Random Number Generator process is run, the resulting output numbers will be used to select the S-BOX that will be used by applying it to be the index of the set of S-BOX provided, so that one S-BOX is selected to be used in the encryption and decryption process. The last process is the main encryption or decryption process, this process we used standard AES calculation with a given selected S-BOX from the RNG process.

3 Results and Discussion

3.1 Results

The experimental data used in this study are in Table 1. Each data is processed using standard AES algorithm and the proposed algorithm is AES with dynamic S-BOX. The type of algorithm that will be used is AES 256-bit with ECB mode. All data will be encrypted which then the results will be decrypted to be able to pull the data from each process. Testing on the S-BOX will be carried out using two methods, namely nonlinearity and SAC which are analyzed on each data.

3.1.1 Abbreviations and Acronyms

In the experiment using the standard AES algorithm, it was found that the average encryption time was 182804.08 nanoseconds or 0.000182804 seconds, while the average decryption time was 143717.8 nanoseconds or 0.000143717 seconds, while for nonlinearity and SAC testing, it was at fixed numbers, namely 112 and 0.504. Details of the experimental results with the standard AES algorithm can be seen in Table 2.

Table 2. Test result on standard AES Algorithm

Test Data	Encryption Time (*nanosecond*)	Decryption Time (*nanosecond*)	Nonlinearity	SAC
1	279621	199284	112	0,504
2	212704	181211	112	0,504
3	174342	187188	112	0,504
4	158868	551851	112	0,504
5	185204	98653	112	0,504
6	297180	173580	112	0,504
7	184586	95493	112	0,504
8	474307	106588	112	0,504
9	109523	99334	112	0,504
10	105142	189723	112	0,504
11	119192	95151	112	0,504
12	**105114**	93151	112	0,504
13	110799	150760	112	0,504
14	148593	102220	112	0,504
15	248217	119277	112	0,504
16	141940	129277	112	0,504
17	145628	81707	112	0,504
18	239863	82509	112	0,504
19	166969	83264	112	0,504
20	147598	**77994**	112	0,504
21	174232	124179	112	0,504
22	181594	86817	112	0,504
23	156969	101950	112	0,504
24	127451	97956	112	0,504
25	174466	283828	112	0,504

3.1.2 *Standard AES*

The experimental scheme using the AES algorithm with dynamic S-BOX is almost the same as testing using standard AES, which records the S-BOX index used, encryption time, decryption time, nonlinearity test results and SAC testing. The complete results of the experiment can be seen in Table 3. In the test results, it was found that the average encryption time was 215431.44 nanoseconds or 0.000215431 seconds, and the average decryption time was 173069.96 nanoseconds or 0.000173069 seconds, the results of the nonlinearity test with an average value of 112 and SAC with an average of 0.499. Tests with the best data are marked in bold.

Table 3. Test result on AES with Dynamic S-BOX

Test Data	S-BOX Index	Encryption Time (*nanosecond*)	Decryption Time (*nanosecond*)	Nonlinearity	SAC
1	6	383484	228521	112	0,498
2	6	302287	196474	112	0,498
3	3	212795	196324	112	0,498
4	6	228296	237653	112	0,498
5	7	154888	116464	112	0,497
6	5	155354	127665	112	0,494
7	1	212142	170655	112	0,501
8	1	235023	100969	112	0,501
9	0	125295	134683	112	**0,508**

10	2	**114154**	241322	112	0,499
11	0	118104	124075	112	**0,508**
12	1	115387	107166	112	0,501
13	4	117444	222992	112	0,497
14	7	163126	263976	112	0,497
15	8	234681	166973	112	0,494
16	0	209436	181157	112	**0,508**
17	6	138039	125426	112	0,498
18	8	559046	105056	112	0,494
19	0	184918	230561	112	**0,508**
20	5	203974	**84027**	112	0,494
21	4	216807	150810	112	0,497
22	2	166163	109285	112	0,499
23	7	289289	177453	112	0,497
24	3	349870	182850	112	0,498
25	8	195784	344212	112	0,494

3.2 Discussions

3.2.1 *Algorithm Performance*

From the algorithm design, by increasing the Random Number Generator process in the proposed algorithm, it is expected that the encryption time performance of the AES algorithm with dynamic S-BOX will decrease from the standard AES algorithm. AES algorithm with dynamic S-BOX has a performance level of 84.85% from standard AES for the encryption process and 83.04% for the decryption process.

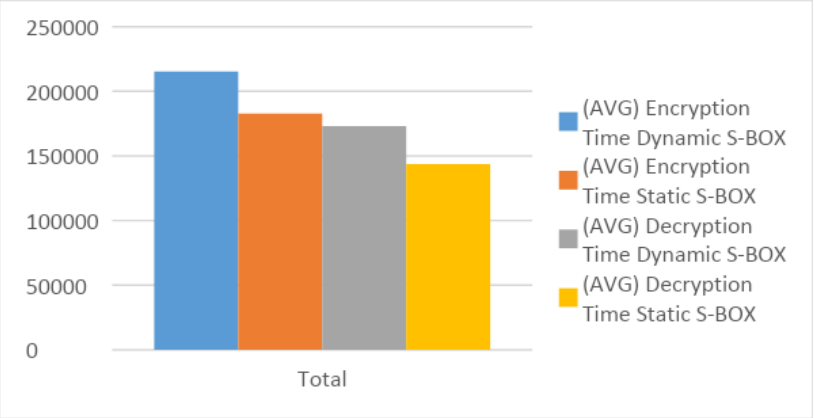

Figure 6. Testing execution time result

3.2.2 *Algorithm Security*

Using dynamic S-BOX or S-BOX that is not a standard S-BOX in the AES algorithm has been proven to increase security in maintaining the security of encrypted information, as has been cited in the explanation of point 2.2.11 and various other studies related to dynamic S-BOX. The S-BOX used by AES uses S-BOX with a 16x16 matrix, with this it is found that there is a combination of 256 against 256 numbers, with this variation it will be very difficult to solve the algorithm without knowing the S-BOX used. However, the formation of S-BOX must meet a minimum of nonlinear criteria so that it will not be easy for crypto-analysts to know.

Therefore, in this study, the S-BOX used was tested first to meet the nonlinear criteria, which Alamsyah (Alamsyah et al., 2018) explained that the resulting S-BOX (used in this study) has a better security level than the S-BOX. BOX in another study. Basically, the proposed algorithm can use any S-BOX as long as it meets the criteria of a safe and non-linear S-BOX, which will automatically increase the security of the algorithm used.

In table 2 we can conclude that the performance of dynamic S-BOX in terms of nonliarity is the same as that of static/standard S-BOX from AES. However, when viewed from Figure 7, the average dynamic S-BOX has an SAC value of 0.499. As for the static S-BOX, it has a SAC of 0.504. The SAC value will get better the closer it is to 0.500. Thus the SAC test on the dynamic S-BOX is better than the SAC value on the static S-BOX with an increase of 0.003 in the SAC value between the two algorithms.

Table 2. Nonlinearity test result

#	*Nonlinearity*	
	Static S-BOX	**Dynamic S-BOX**
Max.	112	112
Min.	112	112
Avg.	112	112

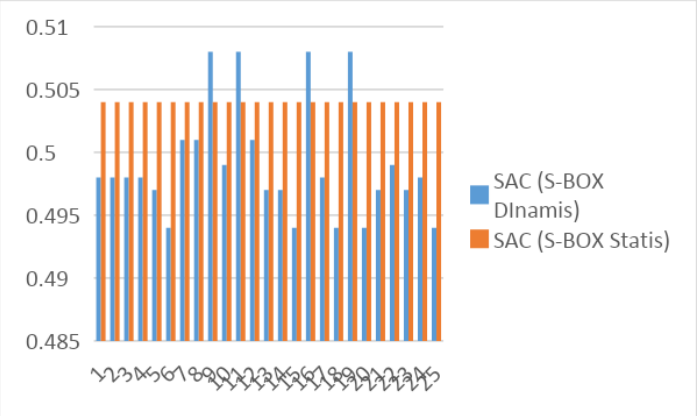

Figure 7. SAC security test result

4 Results and Discussion

Based on the research conducted, the conclusion obtained is that the use of dynamic S-BOX in the AES algorithm can be applied by developers and can increase the security of the resulting ciphertext. The total dynamic S-BOX that can be used from the developed algorithm is very flexible so that it will increase the variety of ciphertext generated by using the RNG to select the index of the S-BOX to be used. The results of security testing on the S-BOX also show an increase from the standard AES S-BOX, namely in the SAC test where the average SAC value of the proposed S-BOX is 0.499 while the standard AES algorithm has an average value of 0.504. The SAC S-BOX value proposed is better because the SAC value will be better the closer it is to 0.500.

References

Alamsyah., Bejo, A., & Adji, T. B. (2018). The replacement of irreducible polynomial and affine mapping for the construction of a strong S-box. *Nonlinear Dynamics*, *93*(4), 2105–2118.

Alsalami, Y., Yeun, C. Y., Martin, T., & Khonji, M. (2016). Linear and differential cryptanalysis of small-sized random (n, m)-S-boxes. *11th International Conference for Internet Technology and Secured Transactions (ICITST)*, 447–454.

Al-Wattar, A. H., Mahmod, R., Zukarnain, Z. A., & Udzir, N. I. (2015). A new DNA-based S-Box. *International Journal of Engineering & Technology*, *15*(4).

Arrag, S., Hamdoun, A., Tragha, A., & Khamlich, S. E. (2013). Implementation of stronger AES by using Dynamic S-Box dependent of Master Key. *Journal of Theoretical and Applied Information Technology*, *53*(2), 196–204.

Daemen, J., & Rijmen, V. (1998). *Aes proposal: Rijndael*. NIST.

Hallappanavar, V. L., Halagali, B. P., & Desai, V. V. (2014). Efficient implementation of AES by modifying S-Box. *IOSR Journal OfComputer Science (IOSR-JCE)*, 35–39.

Ibrahim, A. A. (2017). Perancangan pengamanan data menggunakan algoritma AES (Advanced Encyption Standard). *Jurnal Teknik Informatika*, *3*(1), 53–60.

Jacob, G., Murugan, A., & Viola, I. (2015). Towards the generation of a dynamic key-dependent S-box to enhance security. *Cryptology EPrint Archive*.

Juremi, J., Mahmod, R., Zukarnain, Z. A., & Yasin, S. M. (2017). Modified AES s-box based on determinant matrix algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering*, *7*(1), 110–116.

Manjula, G., & Mohan, H. S. (2016). Constructing key dependent dynamic S-Box for AES block cipher system. *2nd International Conference on Applied and Theoretical Computing and Communication Technology (ICATccT)*, 613–617.

Munir, R. (2004). Pengantar Kriptografi. In R. Munir (Ed.), IF5054 Kriptografi (1-11). Institut Teknologi Bandung