Performance Analysis of Advanced IoT Encryption on Serialization Concept: Application Optimization Approach

Johan Setiawan[1], Muhammad Taufiq Nuruzzaman[2]*
Department of Informatics
Universitas Islam Negeri Sunan Kalijaga Yogyakarta
Yogyakarta, Indonesia

***Abstract*—This study investigates the effect of serialization concepts with cipher algorithms and block mode on structured data on execution time in low-level computing IoT devices. The research was conducted based on IoT devices, which are currently widely used in online transactions. The result of overheating on the CPU is fatal if the encryption load is not reduced. One of the consequences is an increase in the maintenance obligations. So that from this influence, the user experience level will have bad influence. This study uses experimental methods by exploring serialization, ciphers, and block mode using benchmarks to get better data combination algorithms. The four test data groups used in benchmarking will produce an experimental benchmark dataset on the selected AES, Serpent, Rijndael, BlowFish, and block mode ciphers. This study indicates that YAML minify provides an optimal encryption time of 21% and decryption of 27% than JSON Pretty if an average of the whole test is taken. On the other hand, the AES cipher has a significant effect on the encryption and decryption process, which is 51% more optimal for the YAML minify serialization**

***Keywords*—Internet of Things; benchmark; cipher; block mode; serialization**

I. INTRODUCTION

Computers and IoT are useful in assisting the activities of certain individuals or business groups. These business domains include Trade, Transportation, Health [1]–[4], and other specific matters discussed in research [5]–[7]. With the expansion of computers, all devices equipped with a microprocessor have been embedded to sustain mobility and the device's toughness. Devices controlled automatically or remotely are a family of IoT (Internet of Things) supporting devices. IoT uses the M2M (machine-to-machine) concept communication[8] without any human relationship [2][9].

Communication between IoT devices uses information data and instructions that have been designed or regulated by the manufacturer. The information sent and received by devices usually does not want to be known or understood by parties or devices [10][4][11][12] with no interest in destroying or converting the information. Therefore, manufacturers should consider durability and safety at a low cost [13]. The information security risks can be in the form of modifications or interruptions, and these risks can affect the continuity of the process or business flow that is running[8][12][14]. In tackling these threats, data encryption is required [15]. Encryption is a method used to convert original data into artificial data to become rugged and not accessible for humans to read. The encryption process's drawback tends to impose more processing on the microprocessor embedded in an IoT device. It can result from small and limited microprocessor capabilities [16][17] and large amounts of data in the encryption process [18]–[20]. As a result of an encryption algorithm's complexity, the microprocessor on the IoT device is more burdened.

The direct effect of microprocessor devices that get high loads or pressures to overheat is the length of the computation process of a device so that it affects UX (User Experience) because it can reduce the level of efficiency [21][22] Users will feel bored in waiting for computation so that it has an impact on ongoing business processes [21][22][23][24]. On the other hand, the impact of overheating microprocessor pressure is that the device is not durable. It harms device providers that have to carry out more routine maintenance. In [3] research discussed one method of encrypting data with a Catalan object base and two structural combinations on IoT devices; however, that study did not discuss the concept of serialization in structured data to the encryption process. Therefore, The research related to the analysis and evaluation of several algorithms that are often used in data encryption which includes; AES, Rijndael, Serpent and Blowfish. The encryption process uses several different data serialization concepts to improve application performance on IoT systems thus that it can provide less computation, time and memory and provide a better impact on user UX (User Experience) while sustaining a level of security information. At the same time, the advantages for companies that will be gained from this article are used as an option for IoT device providers in dealing with the problem of overheating on the low-level computational microprocessor utilized.

The rest of this article is organized as follows. Section II discusses previous research that has been carried out as a literature study for the author. Section III discusses the research methods used. The experiment is a method used by the author to obtain benchmark data on a combination of serialization, cipher and block mode. Section IV provides data design, benchmark flow design, data collection process and analysis process from experiments. Section V concludes this article.

*Corresponding Author

II. Related Works

Data is an essential thing in business that must be secured when transmitted over public networks. It relates to attacks that threaten data modification and interruption. Encryption and data authentication schemes that are implemented can shield data from these attacks to not be read by unauthorized people [4][10]. Nevertheless, on the other hand, encryption can burden the microprocessor [3], resulting in overheating. Sudip Maitra et al. Have published research related to evaluating the performance of the encryption algorithm on IoT devices with the XTEA and AES algorithms to obtain an algorithm with more optimal memory, time, and energy. The research has been conducted using an experimental method utilizing an oscilloscope device to help identify the energy consumed in the encryption process. From these results, it was found that the XTEA algorithm is better in terms of efficiency if the IoT device does not use the AES accelerator [18]. On the other hand, Geovandro C. et al. Has researched the evaluation of cryptographic algorithms' performance on IoT and operating systems [17].

Nur Rachmat et al. has implemented the analysis of the performance of Rijndael, Serpent, Twofish on an android smartphone. The conclusion of the research that Serpent has good performance at execution time [25]. Furthermore, Muzafer H. Saračević et al. have been carried out providing a proposal to use encryption on Catalan object-based IoT and two structural combinations. In this research, the whole procedure is based on the Catalan number's characteristics and the representation numbers and combinatorial problems. Apart from improving the quality of encryption, that study recommended lightweight computing like e-health and smart cities [3]. Moreover, several studies are almost similar in [26]–[28].

III. Research Method

Researchers use the research method to collect data that followed up as material for investigation and analysis. The research method provides information from the research design to be composed of time, place, and data source. At this condition, the researcher applies experimental research methods. It will provide an overview of the effect of data serialization of pretty JSON, JSON minify, YAML, and YAML minify as machine-to-machine communication. It is against several encryption algorithms/ciphers at AES, Blowfish, Rijndael, and Serpent on IoT devices to get better device performance. The steps taken in this study were coding, data collection, data grouping, benchmarking, and the analysis process. The procedure of the research method used in this study can be seen in Fig. 1.

A. Research Tools

Tools are vastly crucial in research since it can affect the results of the analysis to be performed. This study uses an IoT device in the form of Orange Pi Zero. It is a tiny computing device with a more complex system to assist the data collection process with specifications shown in Table I.

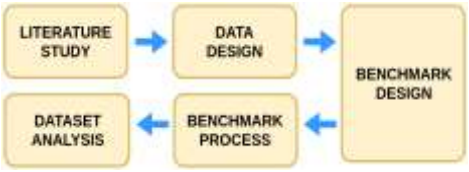

Fig. 1. Research Method.

TABLE I. IoT Hardware Specification

Type	Specifications
CPU Manufacture	AllWinner / ARMv7
CPU Core	4 Cores
CPU Speed	1.5 GHz
RAM	512 MB
Disk	32 GB
OS	Linux Ubuntu 18.04 LTS

B. Data Collection

Data will operate as serialized data in JSON and YAML with the pretty and minify schemes in this process. From some of these data serialization concepts, the next step is to benchmark the data encryption process to adjust the system or algorithm to get better processing on specific platforms [29]. The benchmarking process in this study uses the Golang programming language. The author uses that in benchmarking to get data benchmarks with low-level programming languages [30]. Thus the golang application will be compiled into binary so that it becomes faster [31].

C. Data Grouping

The grouping process will determine the percentage level of time efficiency or processing carried out in the encryption and decryption process. The process will group on the type of encryption or decryption used based on data serialization. From the benchmark results obtained, researchers can compare these results. A more efficient variety of data serialization, cipher, and block mode combination will be discovered for the IoT application encryption and decryption process.

IV. Result and Discussions

A. Benchmark Design

Several schemes of the data structure to be used in the benchmark have different characteristics so that the data used will have various levels of influence. The data sample is traditional data commonly used in communication between devices. The traditional JSON data structure is used as a variety of file sizes, including 1.5 KB, 2.1 KB, 3.0 KB and 3.5 KB. The data will be converted with serialization concept.

At this step, some of the cipher and block mode algorithms listed in Table II will be designed as a benchmark process stage executed after the data serialization process. In this case, it is assumed that the data have been through the serialization process such as JSON and YAML with the concept of minifying and pretty. The process diagram can be seen in Fig. 2.

TABLE II. CIPHER AND BLOCK MODE

Cipher	Block Modes
AES-256	CBC, CTR, ECB
BlowFish	CBC,EBC,OFB
Serpent	CBC,ECB,OFB,NOFB,CFB,NCFB,CTR
Rijndael-256	CBC,ECB,OFB,NOFB,CFB,CTR,NCFB

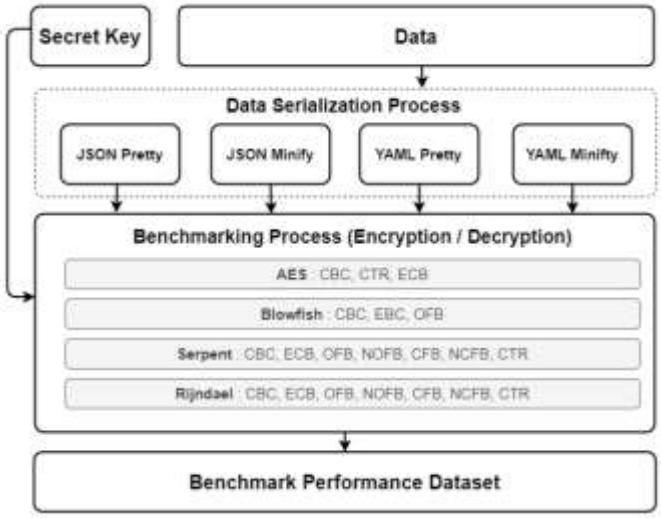

Fig. 2. Benchmarking Flow.

In Fig. 2, there are data arrays and secret keys that have been provided. The data is used in the serialization process and converted into an appropriate data structure. Furthermore, in the benchmarking process section, the data used is data that has gone through serialization. In conducting benchmarks, serialized data will be looped with the number of *n* in the benchmark function algorithm and block mode using the provided secret key. After the benchmarking is complete, the performance dataset will be used to conduct research using comparative analysis.

In Table III, Schemes 1 and 2 are schemes that are often operated in serialized data encryption and usually used in applications. Meanwhile, Schemes 3 and 4 are comparison schemes using different data serialization concepts and rarely used from traditional types. With these functions' combination, the benchmark process will be carried out to gain speed/application optimization on the IoT encryption process.

B. Experiment Flow Design

In directing experiments, the Package Benchmark used by the researcher used a package that was already installed in the Golang programming language. In contrast, the operating system's encryption package is libmcrypt-dev, which is also embedded in the Operating System used to have a better effect on library performance. Furthermore, the encrypt library used in the project is able to be seen in the repository (*https://github.com/mfpierre/go-mcrypt*). In preparing the benchmark function, the researcher provides a specimen that can be seen in Fig. 3.

TABLE III. COMBINATION OF ENCRYPT AND SERIALIZATION

Schemes	Functions
Scheme 1	EncryptAlg(JSON(Data))
Scheme 2	EncryptAlg(Minify(JSON(Data)))
Scheme 3	EncryptAlg(YAML(Data))
Scheme 4	EncryptAlg(Minify(YAML(Data)))

```
cpu profile | mem profile | run benchmark | debug benchmark
func Benchmark_Encrypt_ChipperName_BlockMode(b *testing.B) {
    for i := 0; i < b.N; i++ {
        EncryptAlg(string(Passphrase), string(Data))
    }
}

cpu profile | mem profile | run benchmark | debug benchmark
func Benchmark_Decrypt_ChipperName_BlockMode(b *testing.B) {
    for i := 0; i < b.N; i++ {
        DecryptAlg(string(Passphrase), string(Data))
    }
}
```

Fig. 3. Benchmark Function Scheme.

In Fig. 3, there is a function with the prefix 'Benchmark' which serves as a marker that the function is applied to perform benchmarking tests against "ChipperName" and "BlockMode." The 'testing' package determines the number of iterations performed in the function. At the same time, 'EncryptAlg' is a function been arranged as an application helper according to the list in Table II and has been adjusted in Table III assuming the 'Data' variable has been serialized first or in 'DecryptAlg' variable 'Data' is data that have been encrypted.

C. Benchmark Result

The purpose of the benchmark process is to find the average execution time of each algorithm on the cipher and block mode used in encryption. The results of this benchmark will then be analyzed in the data analysis step. The formula for the benchmark calculation of all sample data is as follows.

$$f(x,y) = \sum_{k=1}^{n} x\big(y(Sk)\big) \quad (1)$$

Where in that function, *x* is the benchmark function used in Fig. 3. *y* is the data serialization function. *n* is total sample data, in this case, the researcher uses four data samples that have been described, and *S* is the list of sample data used and has been used to serialize. From the results of calculations using the formula 1, the data to be obtained will be listed as in Tables IV and V on each scheme, cipher and block mode used.

In Table IV and V to measure the percentage of time reduction from scheme 1 with scheme 4 uses the formula which can be seen in formula 2 - 4.

$$j(x) = \sum_{k=1}^{n} x\big(b(Sk)\big) \quad (2)$$

$$y(x) = \sum_{k=1}^{n} x\big(m(Sk)\big) \quad (3)$$

$$f(x) = \frac{j(x) - y(x)}{j(x)}\ x\ 100 \quad (4)$$

TABLE IV. BENCHMARK RESULT OF THE ENCRYPT COMBINATION WITH SERIALIZATION

No.	Cipher	Block Mode	Total Time (*ns*)				Reduction Time from Scheme 1 to 4 (%)
			Scheme 1	*Scheme 2*	*Scheme 3*	*Scheme 4*	
1	AES-256	CBC	1253844	667272	680424	628043	49,91
2	AES-256	CTR	1805709	1005851	1008796	979993	45,73
3	AES-256	ECB	768435	388808	394922	358955	53.29
4	Blowfish	CBC	4034594	3782520	3778960	3752085	7.00
5	Blowfish	EBC	4130058	3887133	3907471	3867405	6.36
6	Blowfish	OFB	6331005	5072219	5072810	4970101	21.50
7	Serpent	CBC	3108647	2831654	2806282	2773793	10.77
8	Serpent	ECB	3251809	3589981	2934591	2886946	11.22
9	Serpent	OFB	10200276	5750340	6385557	6067675	40.51
10	Serpent	NOFB	3501421	3661894	3163832	3116317	11.00
11	Serpent	CFB	9883248	5502608	6062519	5742311	41.90
12	Serpent	NCFB	3439166	3057695	3118968	3055999	11.14
13	Serpent	CTR	3293048	2998750	2973943	2931130	10.99
14	Rijndael-256	CBC	3526985	3146207	3130867	3071622	12.91
15	Rijndael-256	ECB	3643133	5075309	3245767	3207467	11.96
16	Rijndael-256	OFB	23662065	11162439	13070587	12081937	48.94
17	Rijndael-256	NOFB	3903236	5163250	3452525	3411664	12.59
18	Rijndael-256	CFB	23303105	10860264	12758767	11777678	49.46
19	Rijndael-256	CTR	3702905	3304169	3273811	3232372	12.71
20	Rijndael-256	NCFB	3831561	3252175	3388191	3347123	12.64

TABLE V. BENCHMARK RESULT OF THE DECRYPT COMBINATION WITH SERIALIZATION

No.	Cipher	Block Mode	Total Time (*ns*)				Reduction Time from Scheme 1 to 4 (%)
			Scheme 1	*Scheme 2*	*Scheme 3*	*Scheme 4*	
1	AES-256	CBC	1258442	656637	656176	607571	51.72
2	AES-256	CTR	1829366	962004	969786	926403	49.36
3	AES-256	ECB	915100	449850	458863	420494	54.05
4	Blowfish	CBC	4301414	3899693	3900940	3875388	9.90
5	Blowfish	EBC	4363084	3986372	3997765	3956421	9.32
6	Blowfish	OFB	6732089	5171293	5194958	5047958	25.02
7	Serpent	CBC	3418887	2886116	2924647	2867847	16.12
8	Serpent	ECB	3496962	3012961	3029684	2971746	15.02
9	Serpent	OFB	10555775	6454063	6522322	6148184	41.76
10	Serpent	NOFB	3731382	3252595	3264580	3220112	13.70
11	Serpent	CFB	10339117	6133240	6175476	5809008	43.82
12	Serpent	NCFB	3769358	3171058	3176920	3136047	16.80
13	Serpent	CTR	3578474	3054691	3066020	3028263	15.38
14	Rijndael-256	CBC	3850404	3216844	3222877	3172646	17.60
15	Rijndael-256	ECB	3917032	3340315	3345704	3305081	15.62
16	Rijndael-256	OFB	24907847	13150971	13183803	12191129	51.06
17	Rijndael-256	NOFB	4226241	3561972	3577534	3524398	16.61
18	Rijndael-256	CFB	24651813	12773199	12888158	11890058	51.77
19	Rijndael-256	CTR	4037440	3365763	3374559	3335614	17.38
20	Rijndael-256	NCFB	4075122	3500502	3500835	3455867	15.20

In this formula, *j(x)* is a function to get the total benchmark value from the JSON serialization data because *b* is a function used for serializing JSON data. *y(x)* is a function to get the total benchmark value from the YAML serialization since *m* was the YAML minify function. In that formula, the value of *n* is given 4 because this value is the total of the data sample. *S* is a list of sample data used. Thus that in calculating the percentage obtained from the value *f(x)*.

D. Analysis

All benchmark data will be grouped and analyzed more deeply; thus, it becomes more informative data in graphical form. In this study, four data samples have been converted into four types of data structures tested on 20 combinations of cipher and block. In this case, the comparison graph is assumed to be derived from the execution time's total evaluation result. A comparison of the traditional scheme 1 against scheme 4 on the cipher used can be seen in Fig. 4 to 11.

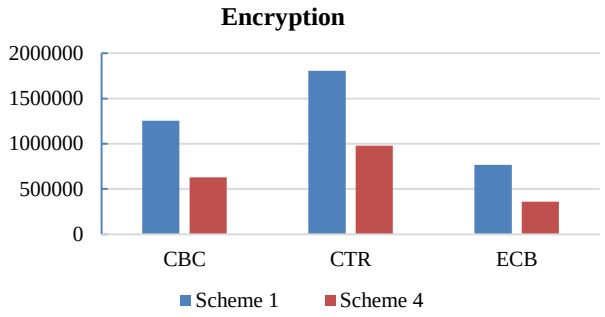

Fig. 4. AES-256 Encryption Time Comparison (*ns*).

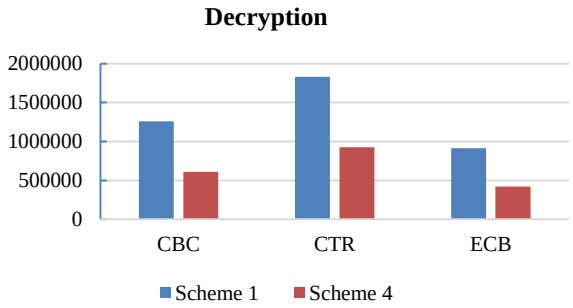

Fig. 5. AES-256 Decryption Time Comparison (*ns*).

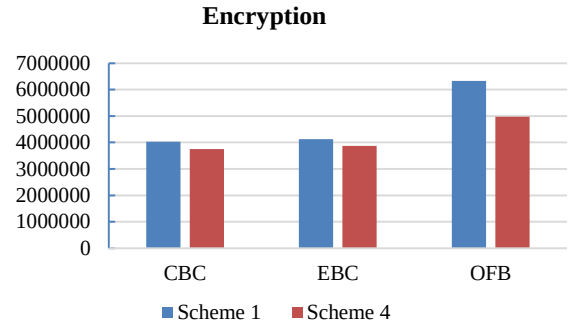

Fig. 6. BlowFish Encryption Time Comparison (*ns*).

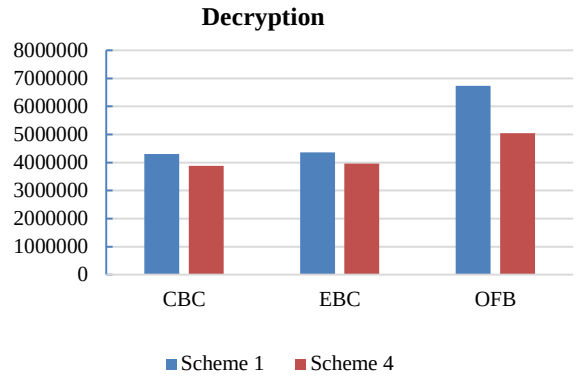

Fig. 7. BlowFish Decryption Time Comparison (*ns*).

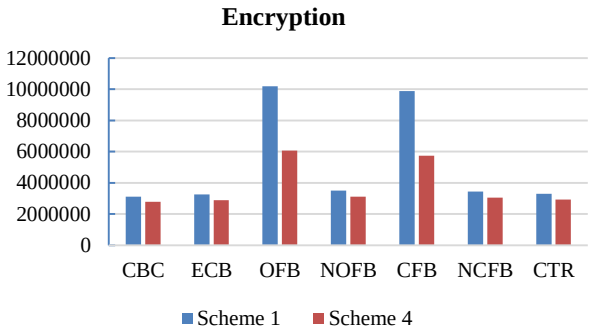

Fig. 8. Serpent Encryption Time Comparison (*ns*).

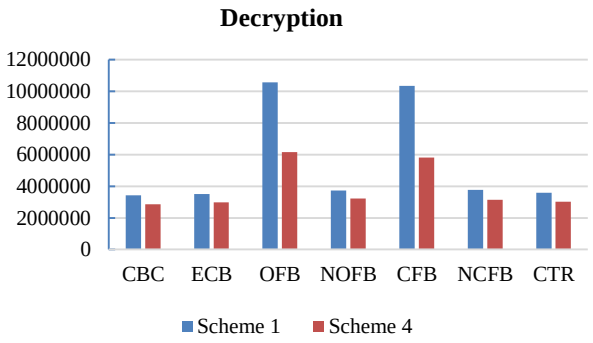

Fig. 9. Serpent Decryption Time Comparison (*ns*).

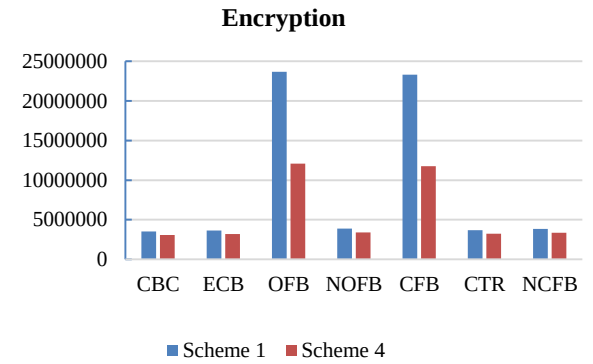

Fig. 10. Rijndael Encryption Time Comparison (*ns*).

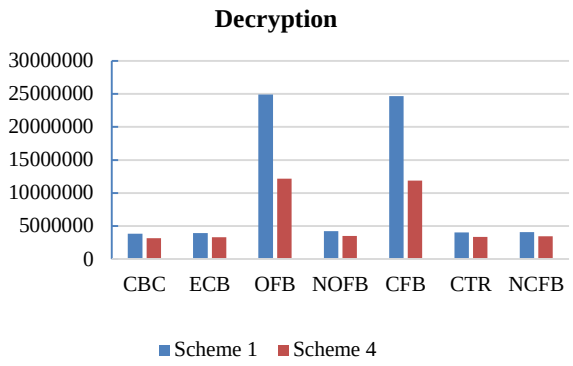

Fig. 11. Rijndael Decryption Time Comparison (*ns*).

Based on Fig. 4 to 11 in the comparison results of schemes 1 and 4 in Table III, the YAML minify serializations method gets an upbeat assessment of the encryption and decryption side. As for the algorithm and block mode used, AES ECB gets better speed than BlowFish, Serpent, and Rijndael. Block modes that provide better performance are ECB, CFB, and OFB. The proposed encryption flow on the low-level IoT platform uses YAML data serialization with the minify scheme. It uses the AES ECB algorithm based on the graphic data described. However, this is very relative to the device used.

V. Conclusion

This research evaluated the cipher and block mode's performance based on several data serialization schemes on low computing devices. The trials' convener carried out using several data serialization. The results were not too significant between scheme one and scheme four on a particular cipher. However, this experiment could have a very significant time-cutting effect on the AES cipher trial with an average of 51% pruning. However, the overall average for encryption will be obtained at 21.85% and 27.36% in decryption. With this research. The hope that it will allow developers to select cipher, block mode, and data serialization to reduce the execution time in the encryption or decryption process.

In the benchmarking process, the author only uses one IoT device. In this case, the author cannot give a definite measure of the figures presented. The author directed a still new system benchmark, and there are no applications that burden the microprocessor. However, it can explain how the serialization, cipher, and block mode combination affects these devices' performance.

The hope for the future, there is further research on this field. For example, by changing data type, data length, protocol, a programming language used or adding the other IoT platform with processor architecture changed.

References

[1] D. Richards, A. Abdelgawad, and K. Yelamarthi, "How Does Encryption Influence Timing in IoT?," 2018 IEEE Glob. Conf. Internet Things, GCIoT 2018, pp. 1–5, 2019.

[2] Y. Hanada, L. Hsiao, and P. Levis, "Smart contracts for machine-to-machine communication: Possibilities and limitations," Proc. - 2018 IEEE Int. Conf. Internet Things Intell. Syst. IOTAIS 2018, pp. 130–136, 2019.

[3] M. H. Saracevic et al., "Data Encryption for Internet of Things Applications Based on Catalan Objects and Two Combinatorial Structures," IEEE Trans. Reliab., 2020.

[4] K. Yelamarthi, M. S. Aman, and A. Abdelgawad, "An application-driven modular IoT architecture," Wirel. Commun. Mob. Comput., vol. 2017, 2017.

[5] H. M. Al-Kadhim and H. S. Al-Raweshidy, "Energy efficient and reliable transport of data in cloud-based IoT," IEEE Access, vol. 7, pp. 64641–64650, 2019.

[6] D. Sharma and D. Jinwala, "Functional encryption in IoT E-Health care system," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9478, pp. 345–363, 2015.

[7] A. D. Dwivedi, L. Malina, P. Dzurenda, and G. Srivastava, "Optimized blockchain model for internet of things based healthcare applications," 2019 42nd Int. Conf. Telecommun. Signal Process. TSP 2019, pp. 135–139, 2019.

[8] P. Radanliev, D. De Roure, C. Maple, J. R. . Nurse, R. Nicolescu, and U. Ani, "Cyber Risk in IoT Systems," Univ. Oxford Comb. Work. Pap. Proj. reports Prep. PETRAS Natl. Cent. Excell. Cisco Res. Cent., vol. 169701, no. 2017, pp. 1–27, 2019.

[9] M. Chen, J. Wan, and F. Li, "Machine-to-machine communications: Architectures, standards and applications," KSII Trans. Internet Inf. Syst., vol. 6, no. 2, pp. 480–497, 2012.

[10] G. A. Utomo, "Ethical hacking," CyberSecurity dan Forensik Digit., vol. 2, no. 1, pp. 8–15, 2019.

[11] M. Stute et al., "A billion open interfaces for Eve and Mallory: MITM, DOS, and tracking attacks on iOS and MACOS through apple wireless direct link," Proc. 28th USENIX Secur. Symp., pp. 37–54, 2019.

[12] I. Fitriani and A. B. Utomo, "Implementasi Algoritma Advanced Encryption Standard (AES) pada Layanan SMS Desa," JISKA (Jurnal Inform. Sunan Kalijaga), vol. 5, no. 3, p. 153, 2020.

[13] M. Weyrich, J. Schmidt, and C. Ebert, "Machine-to-Machine Communication," IEEE, vol. 31, no. 4, pp. 19–23, 2014.

[14] P. Radanliev et al., "Definition of Internet of Things (IoT) Cyber Risk Discussion on a Transformation Roadmap for Standardisation of Regulations Risk Maturity Strategy Design and Impact Assessment," Sensors, no. March, pp. 1–9, 2019.

[15] M. M. Yahaya and A. Ajibola, "Cryptosystem for Secure Data Transmission using Advance Encryption Standard (AES) and Steganography," Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol., vol. 5, no. 6, pp. 317–322, 2019.

[16] M. Khari, A. K. Garg, A. H. Gandomi, R. Gupta, R. Patan, and B. Balusamy, "Securing Data in Internet of Things (IoT) Using Cryptography and Steganography Techniques," IEEE Trans. Syst. Man, Cybern. Syst., vol. 50, no. 1, pp. 73–80, 2020.

[17] G. C. C. F. Pereira, R. C. A. Alves, F. L. da Silva, R. M. Azevedo, B. C. Albertini, and C. B. Margi, "Performance evaluation of cryptographic algorithms over IoT platforms and operating systems," Secur. Commun. Networks, vol. 2017, 2017.

[18] S. Maitra, D. Richards, A. Abdelgawad, and K. Yelamarthi, "Performance Evaluation of IoT Encryption Algorithms: Memory, Timing, and Energy," SAS 2019 - 2019 IEEE Sensors Appl. Symp. Conf. Proc., pp. 6–11, 2019.

[19] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," IEEE Internet Things J., vol. 5, no. 4, pp. 2483–2495, 2018.

[20] M. Botta, M. Simek, and N. Mitton, "Comparison of hardware and software based encryption for secure communication in wireless sensor networks," 2013 36th Int. Conf. Telecommun. Signal Process. TSP 2013, pp. 6–10, 2013.

[21] A. Crescenzi, D. Kelly, and L. Azzopardi, "Impacts of time constraints and system delays on user experience," CHIIR 2016 - Proc. 2016 ACM Conf. Hum. Inf. Interact. Retr., pp. 141–150, 2016.

[22] D. Biduski, E. A. Bellei, J. P. M. Rodriguez, L. A. M. Zaina, and A. C. B. De Marchi, "Assessing long-term user experience on a mobile health application through an in-app embedded conversation-based questionnaire," Comput. Human Behav., vol. 104, 2020.

[23] O. Noguchi, M. Munechika, and C. Kajihara, "A Study on User Satisfaction with an Entire Operation Including Indefinite-Length Response Time ," Total Qual. Sci., vol. 2, no. 2, pp. 70–79, 2016.

[24] L. T. Yong, "User experience evaluation methods for mobile devices," 2013 3rd Int. Conf. Innov. Comput. Technol. INTECH 2013, pp. 281–286, 2013.

[25] N. Rachmat and Samsuryadi, "Performance analysis of 256-bit aes encryption algorithm on android smartphone," J. Phys. Conf. Ser., vol. 1196, no. 1, 2019.

[26] N. Su, Y. Zhang, and M. Li, "Research on data encryption standard based on AES algorithm in internet of things environment," Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019, no. Itnec, pp. 2071–2075, 2019.

[27] H. K. Kim and M. H. Sunwoo, "Low Power AES Using 8-Bit and 32-Bit Datapath Optimization for Small Internet-of-Things (IoT)," J. Signal Process. Syst., vol. 91, no. 11–12, pp. 1283–1289, 2019.

[28] V. K. Sarker, T. N. Gia, H. Tenhunen, and T. Westerlund, "Lightweight Security Algorithms for Resource-constrained IoT-based Sensor Nodes," IEEE Int. Conf. Commun., vol. 2020-June, 2020.

[29] I. I. Conference, "OpBench: A CPU performance benchmark for ethereum smart contract operation code," Proc. - 2019 2nd IEEE Int. Conf. Blockchain, Blockchain 2019, pp. 274–281, 2019.

[30] S. S. Brimzhanova, S. K. Atanov, M. Khuralay, K. S. Kobelekov, and L. G. Gagarina, "Cross-platform compilation of programming language Golang for Raspberry Pi," ACM Int. Conf. Proceeding Ser., vol. Article 10, pp. 1–5, 2019.

[31] C. Wang et al., "Go-Clone: Graph-embedding based clone detector for Golang," ISSTA 2019 - Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal., pp. 378–381, 2019.