

AESHA3: Efficient and Secure Sub-Key Generation for AES Using SHA-3

Ankush Soni* Sanjay K. Sahay† Parit Mehta‡

Abstract

Advanced Encryption Standard (AES) is one of the most widely used symmetric cipher for the confidentiality of data. Also it is used for other security services, viz. integrity, authentication and key establishment. However, recently, authors have shown some weakness in the generation of sub-keys in AES, e.g. bit leakage attack, etc. Also, AES sub-keys are generated sequentially, which is an overhead, especially for resource-constrained devices. Therefore, we propose and investigate a novel encryption AESHA3, which uses sub-keys generated by Secure Hash Algorithm-3 (SHA3). The output of SHA3 is one-way and highly non-linear, and random. The experimental analysis shows that the average time taken for generating the sub-keys to be used for encrypting the data using our approach i.e. AESHA3 is ~ 1300 times faster than the sub-key generated by the standard AES. Accordingly, we find that AESHA3 will be very relevant not only in terms of security but also it will save the resources in IoT devices. We investigated AESHA3 in Intel Core i7, 6th Generation processor and Raspberry Pi 4B and found that up to two MB data encryption is very significant, and lesser the data size, more the resource saving compared to AES.

Advanced Encryption Standard, Sub-keys, Secure Hash Algorithm-3, IoT Devices

1 Introduction

Advanced Encryption Standard (AES) is the most widely used symmetric key cipher, basically for data confidentiality. However it can provide other security services viz. integrity, authentication and key establishment in several real-life applications, including financial services, data centres, web security, etc. The AES has three variant based on the key size i.e. AES-128, AES-192 and AES-256. The design of AES is cryptographically strong [2], and to date, no attack better than brute force has been found. Additionally, because of the 256-bit key variant, it might even cope with the offing quantum computations for the next two decades. However, the sub-key generation process of AES from the master key is arguably frail [10]. Also, with the advent of smart

*Dept. of CSIS, BITS Pilani K.K. Birla Goa Campus, Goa, India, Email: p20180413@goa.bits-pilani.ac.in

†Dept. of CSIS, BITS Pilani K.K. Birla Goa Campus, Goa, India, Email: ssahay@goa.bits-pilani.ac.in

‡Dept. of CSIS, BITS Pilani K.K. Birla Goa Campus, Goa, India, Email: h20210036g@alumni.bits-pilani.ac.in

devices and the Internet of Things (IoT) over the last few years, there is a high demand for an efficient and secure algorithm for these resource and energy-constrained devices. Therefore in this paper we propose a novel efficient and secure sub-key generation for AES using Secure Hash Algorithm-3 (SHA-3) for security services, and named AE-SHA3. We use the same three fundamental layers of AES (Key Addition, Shift Rows and Mix Columns) approved by the National Institute of Standards and Technology (NIST) [2]. However the sub-keys are generated from SHA3, which is highly random, non-linear and one-way (i.e. the key cannot be generated in the reverse order) [6] i.e., it has all the properties that the NIST-approved AES sub-keys possess. Hence, we have used it to generate the sub-keys of all rounds of AES for the encryption of the data.

The remainder of the paper is divided as follows: Section II briefly discusses the related work. Section III gives the brief description of AES and SHA3. While section IV discusses the problem overview and our approach for sub-key generation for AE-SHA3. Section V and VI describes the experimental setup and analysis of our results respectively. Finally, the section VII contains the conclusion of the paper.

2 Related Work

According to Kerckhoff's Principles [13], a cipher or cryptosystem should be secure even if the attacker knows all details about the system, except the secret key i.e. the system should be secure even if the attacker knows the encryption and decryption processes. Therefore, the keys used for any cipher are the most important aspect of achieving the desired security level. In symmetric cipher, generally sub-keys are generated from the selected key so that even if the selected key is weak, the sub-key generation process makes the sub-keys highly random so that security of the cipher entirely lies on the sub-keys. Therefore, time-to-time researchers investigated one of the widely used AES sub-keys generation processes. The first weakness in the AES key schedule was discovered by Lauren and Matt in 2002 [10]. According to them, the key schedule of the AES is vulnerable to bit leakage i.e. even with partial knowledge of the previous sub-key, all the other sub-keys can be generated. They proposed a rectified key schedule in which every subkey depends on every bit of the original key. Their proposed key schedule secures the keys from bit leakage. However, the computational resources required for generating the sub-keys remain more or less the same as the original AES sub-key generation processes. Later in 2008, Bahrak et al. [21] proposed a differential attack which exploits differences at the intermediate state of the AES algorithm and time-memory trade off. They have shown that the best differential attack requires $2^{115.5}$ chosen plain-texts, 2^{109} bytes of memory and seven rounds only to attack AES-128. However, it is almost impossible to get $2^{115.5}$ of plain-texts.

In 2010, Biryukov et al. [26] show that AES-192 and AES-256 security level can be 176 and 199 bits respectively. However, still, it is completely impractical to find the key. The reason for their reduced security level may be due to generating 1.5 and 2 sub-keys with the standard AES key schedule. Therefore, their attack is not applicable to AES-128 because one sub-key is generated in each iteration in AES-128. Also, their attack depends on how the keys are related.

Understanding the importance of light weight cipher for IoT devices, Bogdanov

et al. [25] in 2014 proposed a light weight encryption scheme which uses AES-128 key schedule. The cipher is an online, single-pass authenticated encryption algorithm that supports optional associated data, and its security relies on nonces. Later in 2019 Alasaad et al. [23] claimed that as the S-box used in AES is a static and fixed matrix, therefore, a backdoor can be built into the cipher to exploit the AES. Hence, they proposed a simple key dependent S-box scheme which generates a dynamic S-box for each round of encryption. Their approach uses some bits of the primary key to directly manipulate the standard S-box in such a way that its content is changed but its cryptographic properties are preserved. They have shown that their proposed method strengthens the cipher against certain attacks at the expense of a relatively modest one-time computational procedure during the set-up phase.

In 2020, Leurent et al. [9], proposed a key schedule in which all the sub-keys of the rounds are generated independently of each other, unlike the original AES key schedule, where the sub-keys are generated from a single master key. Their proposed approach security level remains the same as that of the original AES. Also, the number of steps for generating the sub-keys was exactly the same. Later in 2021, G. Leurent et al. [9] proposed a modified AES key schedule in which all the sub-keys of the rounds are generated independently, unlike the original AES key schedule, where the sub-keys are generated sequentially from the master key. Their basic idea is to parallelise the computation of the key schedule. They generate an equivalent representation of the AES key schedule using invariant subspace attacks. However, the number of computations for the variants of AES are the same as the original AES.

Recently, Sawka et al. [14] proposed a scheme to create key expansion algorithms for modern block ciphers. Their approach uses the sponge construction for creating the key expansion algorithm. Further, their method uses the key expansion algorithm on a specifically designed block cipher, IJON. As their approach was specific to the custom block cipher, its scalability is not guaranteed.

3 Brief Description of Advanced Encryption Standard and Secure Hash Algorithm-3

3.1 Advanced Encryption Standard

The first approved block cipher was Data Encryption Standard (DES) [27], which is still secure as per its design, i.e. no attack better than brute force attack has been found yet. However, due to its small key size (56 bits), DES is no longer recommended; instead, one can use 3DES, which is three times slower than DES and provides 112-bit security only. Therefore, to have an efficient and more secure symmetric cipher, NIST approved AES in 2002 [2]. In general, symmetric block cipher processes n bits of plain text and m bits of the key to generate n bits of random cipher text. The security of the modern symmetric cipher is achieved by applying two characteristics, i.e. confusion and diffusion, introduced by Claude Shannon [15] in multiple rounds (in AES, the number of rounds depends on the key size to produce a completely random ciphertext). The confusion property obscures the relationship between the key and cipher text, while diffusion obscures the relationship between plain text and cipher text

Table 1: Number of rounds and sub-keys in the variants of AES

AES Version	AES-128	AES-192	AES-256
Number of bits in the sub-key	128	192	256
Number of rounds	10	12	14
Number of sub-keys	11	13	15
Total number of bits required	1408	1664	1920

to resist cryptanalysis. AES transforms the input data into three layers, i.e. substitution layer for confusion property, diffusion layer for diffusion property and in key addition layer, data state is XORed with the generated non-linear sub-keys to make the cipher random.

The AES operates on bytes rather than bits, and its input (plaintext) of 128 bits is represented as 16 bytes, arranged in a 4 x 4 matrix. This plaintext of the matrix is known as the initial state and is modified as the algorithm progresses. AES comes in three different variants viz. 128, 192, and 256 input key bits. Each of these is arranged in a 4 x 4/6/8 matrix, each column representing a word of 4 bytes. However, in encryption processes, only 128 bits are used to XOR the data state irrespective of the main key size. Therefore, the number of rounds of encryption processes increases with key size and are 10, 12 and 14 rounds for 128, 192 and 256-bit key sizes, respectively. Table 1 shows the number of rounds, number of sub-keys and total number of bits generated to get the ciphertext. The number of sub-keys is one more than the number of rounds because before the start of the encryption processes, the original plaintext is initially XORed with the first subkey (master-key), and this XORing of plaintext before the start of the actual encryption processes is known as key whitening.

3.2 Secure Hash Algorithm-3

A cryptographic hash function (CHF) is a one-way function which takes arbitrary size input and provides a highly compressed fixed-size output. It is key-less but plays a very important role in digital security, viz., checks the integrity of the data because it is deterministic and highly sensitive (on average, when a single bit in the data is changed, approximately half of the bits in the hash output will be modified), and it is computationally impossible to find another message for the known hash value. Therefore, it is also known as checksum, message digest and fingerprint. The security level of a hash function is determined by computationally in-feasibility to find two different inputs that generate the same hash, i.e. collision resistance, and to find the original input from the hash value, i.e. preimage-resistance. Generally, if n is the size of the hash value, then a good hash function shall have $n/2$ bits security level.

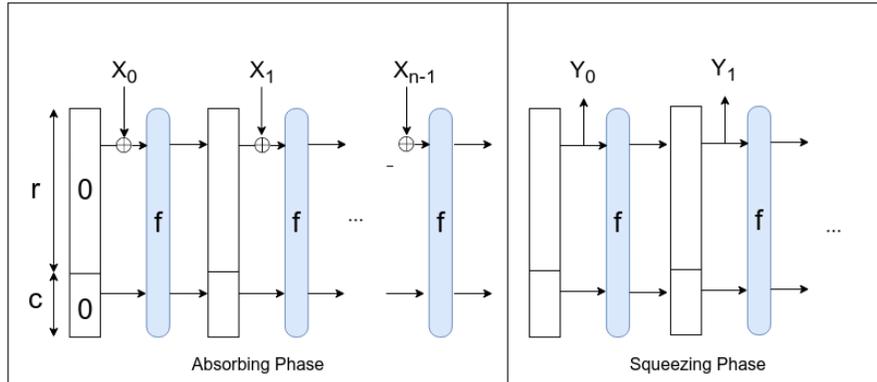


Figure 1: A schematic of SHA-3

The first CHF was designed by Rabin in the late 70s. He proposed a 64-bit hash function using DES block cipher [18]. Later, in the 80s, more hash functions were proposed, and in 90s, MD5 and SHA-1 were used in several applications [6]. However, in 2004, it was shown that finding the collision in Message Digest-5 (MD5) is easy, and the security level of SHA-1 is significantly less than the standard required security. Hence SHA-2 series (SHA-256, SHA-384, SHA-512) has been approved, and to date, no collision has been found in SHA-2. However, understanding the significance of CHF for security services, in 2012, NIST approved SHA-3 [6] to meet out the digital security requirements. The design of SHA-3 is very different from earlier hash functions. A schematic of the SHA-3 is shown in figure 1, which is based on sponge construction. It has two phases: in the first phase, the message is absorbed into the sponge, and in the second phase, the result is squeezed. In the absorbing phase, message blocks (x_i) are XORed into a subset of the state and the output blocks are read from the same subset of the state and alternated with the state transformation function (f). The size of the part of the message that is written and read is known as the *rate* (r), and the size of the part that is untouched by input/output is known as the *capacity* (c). The capacity determines the security of the SHA-3 and is $c/2$, and the security level does not change even one take more than the double the bit of the desired security level from the output of the SHA-3. The message transformation is done by the f -function in five steps called θ , ρ , π , χ and ι in which computation of the parity, bit-wise rotation, permutation of 25 words, bit-wise combination along rows and exclusive-OR are done respectively. To get the hash, after the absorption is completed, the function f of the state is taken until the required length is obtained, i.e. $\text{Hash}(x) = Y_0 || Y_1 || \dots$

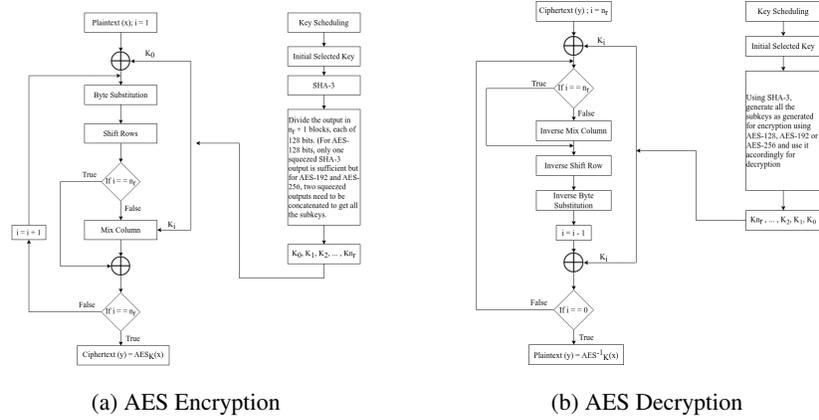


Figure 2: Flowchart for AES

4 Problem Overview and Proposed Approach

4.1 Problem Overview

In AES, the round sub-keys are generated sequentially from the master key using a non-linear g -function, which uses the same S-box that is used in the byte substitution layer [2]. Depending on key size, the number of words (32 bits) are generated. This sequential generation of keys creates additional computational overhead, resulting in more resource utilisation, especially in resource-constrained IoT devices. Also, the authors [2, 9, 16, 17] pointed out the weakness in the key generation process of the AES. Therefore, we investigate SHA-3 for efficient and secure sub-key generation processes for data encryption and decryption.

4.2 Proposed Approach

We propose and investigate a novel and efficient approach for generating the sub-keys using SHA-3. The output of the squeezed phase in SHA-3 provides a non-linear one-way 1600-bit output and can be used as sub-keys in the encryption of data in AES. The required number of bits for encrypting data in different variant of AES is shown in Table 1. For AES-128, just first output of the squeezing phase is sufficient to have all the 11 sub-keys. However, in case of AES-192 and AES-256, one more iteration of the squeezing phase had to run to get all the sub-keys to encrypt the data, i.e. one can have 3200 bits of non-linear bits to make 13 or 15 sub-keys, i.e.

$$\text{Hash}(x) = Y_0 || Y_1$$

where x denotes the input string, Y_0 and Y_1 are the output of SHA-3 squeezing phase, which is of 1600 bits. The processes of squeezing phase in such that from Y_1 it is computationally infeasible to find Y_0 and taking any sequence of 128 bits from Hash(x)

does not effect the security of the sub-keys. Figures 2a and 2b show the flowchart of AES encryption and decryption processes of AESHA3.

5 Experimental Setup

To investigate the proposed novel approach, we implemented it on Intel Core i7 6th generation, 12 GB RAM, in Ubuntu 22.04 LTS OS and Raspberry Pi 4, 8 GB RAM with Raspbian OS using Python 3.10. First, we run the SHA-3 10,000 times with random inputs strings and found the average time taken for generating the sub-keys to be used for encrypting the data using the AES three layer is ~ 1300 times faster than sub-key generated by the standard AES. The detail results are shown in table 2. Then we used Electronic Code Book modes of operations i.e. encrypting data of 128 bits or more independently to find how much time can be saved for data encryption for IoT devices, which are generally resource constrained.

Table 2: Time taken in milliseconds for the sub-key generation by AES and AESHA3

System Specifications	Operating System	AES-128	AESHA3-128	AES-192	AESHA3-192	AES-256	AESHA3-256
Intel Core i7 6th generation CPU	Ubuntu 22.04 LTS	1334.31	1.21	1403.26	1.231	1442.18	1.237
Raspberry Pi 4B 8GB RAM	Raspbian OS	5996.36	4.55	6084.61	4.74	6205.45	4.69

Table 3: Time taken and efficiency to encrypt the data up to 16 MB using AES and AESHA3 in Intel Core i7 6th Generation processor.

File Size	Total Time AES-128 (ms)	Total Time AESHA3-128 (ms)	Efficiency of AESHA3-128 (X)	Total Time AES-192 (ms)	Total Time AESHA3-192 (ms)	Efficiency of AESHA3-192 (X)	Total Time AES-256 (ms)	Total Time AESHA3-256 (ms)	Efficiency of AESHA3-256 (X)
1 KB	1347.24	15.68	85.92	1417.22	19.38	73.12	1464.15	22.70	64.5
2 KB	1355.44	23.69	57.21	1424.68	26.26	54.25	1476.69	34.88	42.33
4 KB	1373.93	42.29	32.49	1448.18	49.99	28.97	1509.12	68.89	21.91
8 KB	1413.34	81.60	17.32	1491.14	92.22	16.17	1548.31	106.64	14.52
16 KB	1473.12	141.49	10.41	1566.51	168.31	9.31	1665.26	222.89	7.47
32 KB	1604.57	272.77	5.88	1729.70	330.97	5.22	1861.00	418.86	4.44
64 KB	1881.59	550.00	3.42	2044.98	646.93	3.16	2225.71	783.58	2.84
128 KB	2402.95	1071.30	2.24	2656.92	1257.83	2.43	2903.74	1462.38	1.98
256 KB	3388.58	2056.83	1.65	3892.48	2493.95	1.56	4358.85	2917.59	1.49
512 KB	5461.41	4129.83	1.32	6398.85	4999.23	1.28	7278.61	5836.77	1.25
1 MB	9649.51	8317.80	1.16	12831.64	11432.31	1.12	13094.25	11653.24	1.12
2 MB	17779.87	16448.04	1.08	22338.69	20939.92	1.06	24701.81	23259.94	1.062
4 MB	35315.09	33984.14	1.04	40327.12	38924.27	1.03	57461.65	56015.52	1.025
8 MB	67805.52	66472.24	1.02	80735.64	79330.45	1.02	114207.08	112728.99	1.013
16 MB	134256.81	132920.93	1.01	160978.36	159569.76	1.01	219933.49	218440.19	1.006

6 Results Analysis

As we find that sub-key generated by SHA-3 takes ~ 1300 less time than the sub-key generated by AES. Therefore for IoT devices we investigated that how much it will

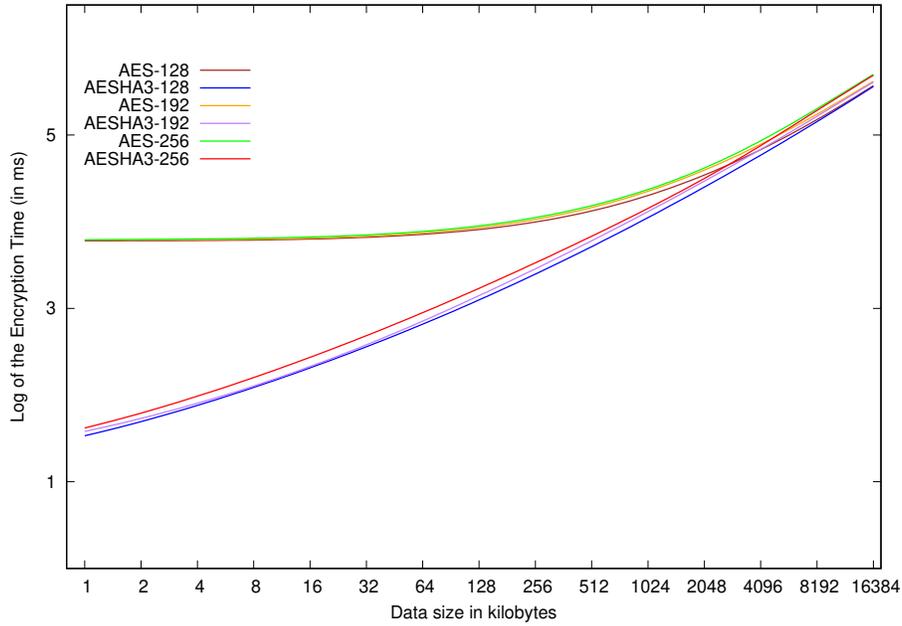


Figure 3: Comparison of the time taken to encrypt the data by AES and AESHA3 by intel core i7, 6th generation processor.

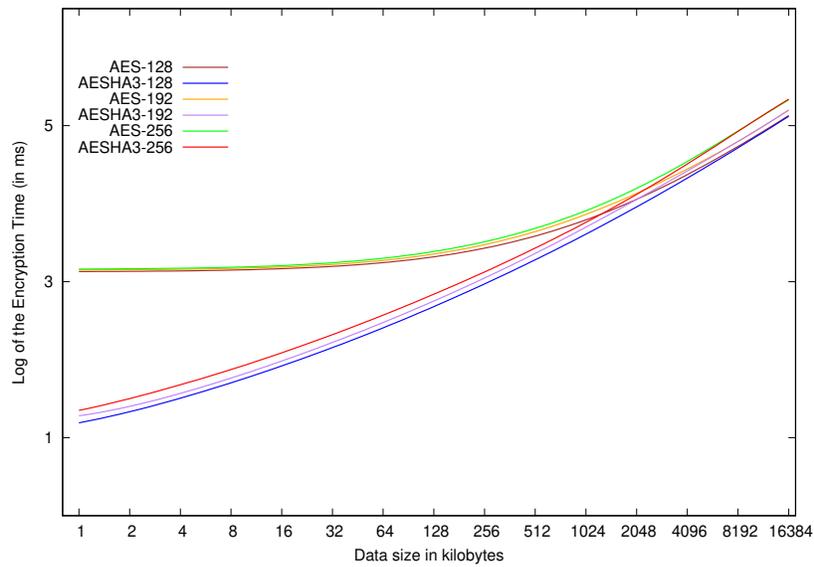


Figure 4: Comparison of the time taken to encrypt the data by AES and AESHA3 by Raspberry Pi 4B.

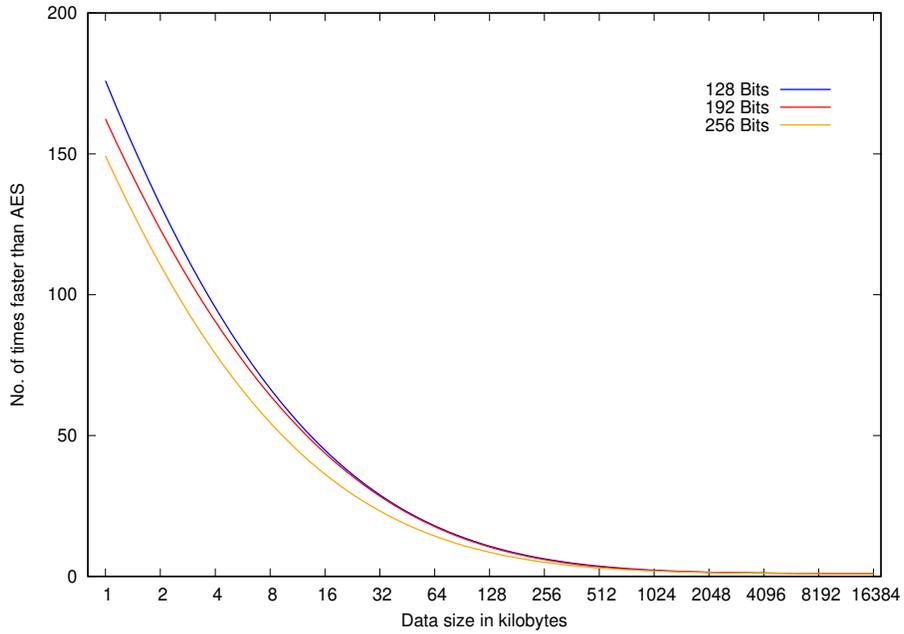


Figure 5: Number of times AESHA3 is faster than AES to encrypt data by Raspberry Pi 4B.

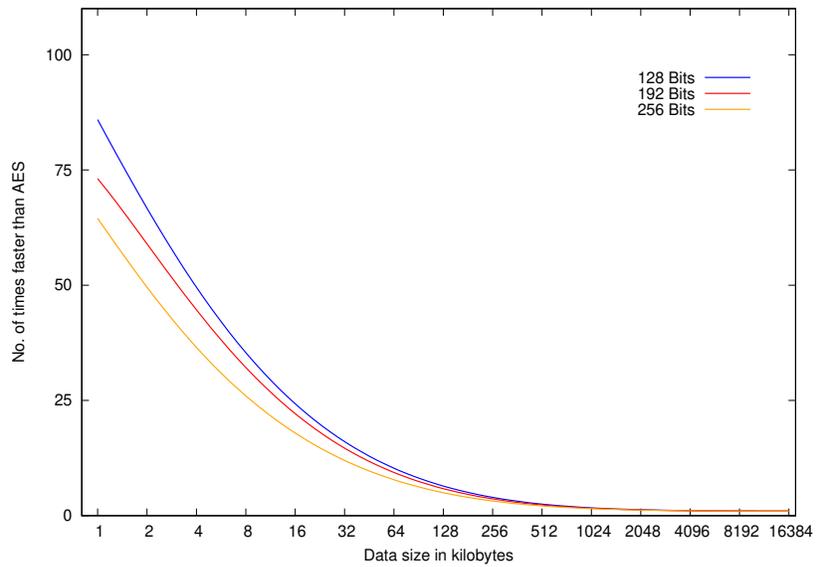


Figure 6: Number of times AESHA3 faster then AES to encrypt data by intel core i7, 6th generation processor.

Table 4: Time taken and efficiency to encrypt the data up to 16 MB using AES and AESHA3 in Raspberry Pi 4B.

File Size	Total Time AES-128 (ms)	Total Time AESHA3-128 (ms)	Efficiency of AESHA3-128 (X)	Total Time AES-192 (ms)	Total Time AESHA3-192 (ms)	Efficiency of AESHA3-192 (X)	Total Time AES-256 (ms)	Total Time AESHA3-256 (ms)	Efficiency of AESHA3-256 (X)
1 KB	6,021.61	34.22	175.97	6,109.35	37.62	162.40	6,230.05	41.73	149.29
2 KB	6,044.45	56.67	106.66	6,131.04	60.83	100.79	6,258.88	71.06	88.08
4 KB	6,088.72	101.13	60.21	6,174.92	103.58	59.61	6,319.90	131.98	47.89
8 KB	6,185.17	197.83	31.27	6,263.59	190.93	32.81	6,445.41	256.86	25.09
16 KB	6,351.80	364.67	17.42	6,445.58	373.80	17.24	6,685.70	497.24	13.45
32 KB	6,700.51	713.00	9.40	6,853.98	782.73	8.76	7,158.61	970.84	7.37
64 KB	7,393.14	1,406.06	5.26	7,722.23	1,651.22	4.68	8,126.02	1,938.22	4.19
128 KB	8,790.51	2,803.58	3.14	9,437.66	3,366.87	2.80	10,078.41	3,891.07	2.59
256 KB	11,656.09	5,668.24	2.06	12,743.25	6,675.07	1.91	13,874.86	7,689.83	1.80
512 KB	17,375.71	11,388.39	1.53	19,583.15	13,510.26	1.45	21,684.56	15,499.15	1.40
1 MB	28,641.21	22,653.57	1.26	32,821.73	26,749.37	1.23	28,572.75	22,384.93	1.28
2 MB	51,266.41	45,279.09	1.13	59,780.93	53,657.67	1.11	67,857.84	61,664.73	1.10
4 MB	98,149.74	92,156.76	1.07	111,105.16	104,965.89	1.06	129,221.49	123,033.96	1.05
8 MB	187,024.09	180,998.08	1.03	218,761.47	212,593.99	1.03	255,063.21	248,797.29	1.03
16 MB	368,645.63	362,567.91	1.02	415,755.80	409,473.02	1.02	499,292.89	493,065.41	1.01

be efficient if data is encrypted by AESHA3. For the purpose first we implemented AESHA3 and AES to encrypt data in intel core i7, 6th generation processor and then we implemented in Raspberry Pi 4B because it has limited computational power and can be considered as IoT device. We encrypted the data in both operating systems starting from 1 KB and doubling the data till 16 MB. We find that up to 2 MB data encryption it is significant, and lesser the data size more the resource saving (table 3 and 4). This is basically due to fact that sub-key generation is one time job. The figure 3 and 4 shows the time taken to encrypt the data with AESHA3 and AES upto 16 MB, and figure 5 and 6 shows the number of times AESHA3 is faster then AES in intel core i7, 6th generation processor and Rasberry Pi 4B respectively.

7 Conclusion

In symmetric cipher, AES is the most widely used cipher for the confidentiality of the data and is also used for the security services viz. integrity, authentication and key establishment. According to Kerckhoff's Principles, a cipher or cryptosystem should be secure even if the attacker knows all details about the system, except the secret key, i.e. the system should be secure even if the attacker knows the encryption/decryption algorithms. However, recently authors have shown some weakness in the generation of sub-keys in AES, hence a threat to the AES cipher. Therefore, we proposed and investigated a novel encryption AESHA3, which uses sub-key generated by SHA3, which provides a one way and highly non-linear output i.e. the overall security will not change. However, our analysis shows that the average time taken for generating the sub-keys to be used for encrypting the data using the three layers of AES is ~ 1300 times faster than sub-key generated by the AES. Therefore, AESHA3 will be very relevant not only in terms of security but also it will save the resources in IoT devices. We implemented AESHA3 in intel core i7, 6th generation processor and Raspberry Pi 4B and find that up to 2 MB data encryption is very significant, and lesser the data size, more the resource saving compared to AES. In this we have made some initial analysis to compare the randomness (i.e. uniformity and independence) of the sub-keys generated by SHA-3 with AES generated sub-keys. The results are encouraging and the detail analysis will be published elsewhere.

References

- [1] Lütkebohle, I. (2021). BWorld Robot Control Software. Available at: <https://bit.ly/3XzbhrU>. [Online; accessed 28-July-2022].
- [2] Dworkin, M., Barker, E., Nechvatal, J., Foti, J., Bassham, L., Roback, E., and Dray, J. (2001). Advanced Encryption Standard (AES). Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD. DOI: <https://doi.org/10.6028/NIST.FIPS.197>.
- [3] Jonsson, J., and Kaliski, B. (2003). Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447, RFC Editor. DOI: 10.17487/RFC3447.
- [4] Moriarty, K., Kaliski, B., Jonsson, J., and Rusch, A. (2016). PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, RFC Editor. DOI: 10.17487/RFC8017.
- [5] Biham, E., and Shamir, A. (2012). Differential Cryptanalysis of the Data Encryption Standard. Springer New York. ISBN: 9781461393146.
- [6] (NIST), N. and Dang, Q. (2012). Secure Hash Standard (SHS), Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, [online], <https://doi.org/10.6028/NIST.FIPS.180-4>.

- [7] Paar, C., and Pelzl, J. (2010). *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer. ISBN: 978-3-642-04100-6.
- [8] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. (2007). Sponge functions. In *ECRYPT hash workshop, 2007(9)*.
- [9] Leurent, G., and Pernot, C. (2021). New Representations of the AES Key Schedule. In *Advances in Cryptology – EUROCRYPT 2021*, pages 54–84. Springer. ISBN: 978-3-030-77870-5.
- [10] May, L., Henricksen, M., Millan, W., Carter, G., and Dawson, E. (2002). Strengthening the Key Schedule of the AES. In *Information Security and Privacy*, pages 226–240. Springer Berlin Heidelberg.
- [11] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and Ferguson, N. (1998). Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15(1), 23–91.
- [12] Anderson, R., Biham, E., and Knudsen, L. (1998). Serpent: A proposal for the advanced encryption standard. *NIST AES Proposal*, 174, 1–23.
- [13] Petitcolas, F. A. P. (2011). Kerckhoffs’ Principle. In *Encyclopedia of Cryptography and Security*, pages 675–675. Springer US. ISBN: 978-1-4419-5906-5.
- [14] Sawka, M., and Niemiec, M. (2022). A Sponge-Based Key Expansion Scheme for Modern Block Ciphers. *Energies*, 15(19), 6864.
- [15] Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4), 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x.
- [16] De Los Reyes, E. M., Sison, A. M., and Medina, R. (2019). Modified AES cipher round and key schedule. *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*, 7(1), 29–36.
- [17] Nikolić, I. (2011). Tweaking AES. In *Selected Areas in Cryptography*, pages 198–210. Springer Berlin Heidelberg. ISBN: 978-3-642-19574-7.
- [18] Preneel, B. (2010). The First 30 Years of Cryptographic Hash Functions and the NIST SHA-3 Competition. In *Topics in Cryptology - CT-RSA 2010*, pages 1–14. Springer Berlin Heidelberg. ISBN: 978-3-642-11925-5.
- [19] Black, J., Cochran, M., and Highland, T. (2006). A Study of the MD5 Attacks: Insights and Improvements. In *Fast Software Encryption*, pages 262–277. Springer Berlin Heidelberg. ISBN: 978-3-540-36598-3.
- [20] Wang, X., Yin, Y. L., and Yu, H. (2005). Finding Collisions in the Full SHA-1. In *Advances in Cryptology – CRYPTO 2005*, pages 17–36. Springer Berlin Heidelberg. ISBN: 978-3-540-31870-5.
- [21] Bahrak, B., and Aref, M. R. (2008). Impossible differential attack on seven-round AES-128. *IET Information Security*, 2(2), 28–32.

- [22] Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., and Shamir, A. (2010). Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In *Advances in Cryptology – EUROCRYPT 2010*, pages 299–319. Springer.
- [23] Alasaad, A., and Alghafis, A. (2019). Key-Dependent S-box Scheme for Enhancing the Security of Block Ciphers. In *2019 2nd International Conference on Signal Processing and Information Security (ICSPIS)*, pages 1–4. DOI: 10.1109/ICSPIS48135.2019.9045900.
- [24] Biham, E., and Keller, N. (2000). Cryptanalysis of reduced variants of RIJNDael. Available at: <https://api.semanticscholar.org/CorpusID:17349612>.
- [25] Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., and Tischhauser, E. (2014). ALE: AES-based lightweight authenticated encryption. In *Fast Software Encryption*, pages 447–466. Springer.
- [26] Biryukov, A., and Wagner, D. (1999). Slide Attacks. In *Fast Software Encryption*, pages 245–259. Springer Berlin Heidelberg. ISBN: 978-3-540-48519-3.
- [27] Biryukov, A., De Cannière, C. (2005). Data encryption standard (DES). In: van Tilborg, H.C.A. (eds) *Encyclopedia of Cryptography and Security*. Springer, Boston, MA. https://doi.org/10.1007/0-387-23483-7_94.