Speeding Up the Internet of Things

LEAIoT: A lightweight encryption algorithm toward low-latency communication for the Internet of Things.

By Muhammad Asif Habib, Mudassar Ahmad, Sohail Jabbar, Syed Hassan Ahmed, and Joel J.P.C. Rodrigues

THE FLOW OF DATA TRAFFIC HAS BEEN VASTLY INCREASED because of the implementation of the Internet of Things (IoT)—and security is a top need for Internet communication. The encryption–decryption process ensures that data access is restricted to legitimate users. But there is a tradeoff between the level of security and the communication overhead.

In this article, a lightweight encryption algorithm toward low-latency communication for the Internet of Things (LEAIoT) is proposed. This algorithm is hybrid in its implementation and achieves data security through an asymmetric encryption algorithm based on a linear block cipher. LEAIoT also uses a symmetric encryption algorithm with the help of a traditional private key. This algorithm was proved to be lightweight because the key generation time was calculated as the lowest in comparison with state-of-the-art encryption algorithms.

Digital Object Identifier 10.1109/MCE.2018.2851722
Date of publication: 9 October 2018

The Data Encryption Standard algorithm is now thought to be insecure for many applications and has many faults.

TWO FORMS OF ENCRYPTION

Cryptographic algorithms have a history as old as civilization. The first known use of cryptography was in ancient Egypt [1]. One eminent example of cryptographic algorithm use is Julius Caesar's implementation of an alphabetic substitution cryptographic system to send secure and protected information to his army [2]. Another prime example is the Hagelin cryptographic algorithm, proposed by Boris Hagelin of Sweden in the 1930s, which was used by the U.S. Army until 1950 [3].

The strength of a cryptographic system depends on the protocol and the length of the key [4]. The application benefits of lightweight encryption are shown in Figure 1. It makes end-to-end communication efficient because it requires fewer resources in terms of computation, memory, and encryption–decryption delay due to its lightweight nature.

SYMMETRIC ENCRYPTION

The expression used to compute the number of symmetric cryptographic required keys is $n(n-1)/2$, where n is the number of nodes or users [5]. The following are the advantages of implementing symmetric key protocols.

- Symmetric key algorithms are faster than asymmetric ones.
- The security level depends on the length of the encryption key.
- The larger the key size, the more difficult it is to break the algorithm because symmetric algorithms perform quite basic mathematical computations on the sequence of bits during the encoding and decoding processes.
- The algorithms do not require a large amount of computing power.

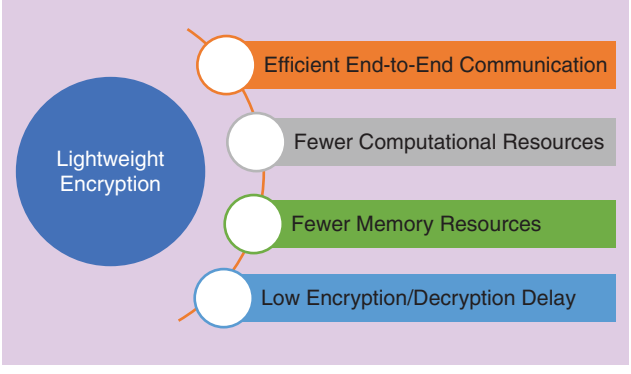

FIGURE 1. The benefits of lightweight encryption.

And here are the disadvantages of symmetric key protocols:

- The symmetric key has no secure mechanism to hand over the secret keys.
- Symmetric key protection and management are challenging.
- The protocols provide confidentiality but no authentication, as the symmetric key is shared.

ASYMMETRIC ENCRYPTION

The public key cryptographic algorithm was first proposed in 1976 [6], [7]. The advantages of asymmetric key systems are as follows.

- Their key distribution schemes are better than those of symmetric cryptography.
- They provide greater scalability than symmetric cryptography.
- They offer confidentiality as well as authentication.

There are also disadvantages.

- Asymmetric key systems work considerably more slowly than symmetric ones.
- They involve mathematically intensive operations and tasks.

This article contributes to the broad spectrum of information in the area of classical encryption and cryptography by developing a novel hybrid encryption model, i.e., the LEAIoT. The Data Encryption Standard (DES) algorithm is now thought to be insecure for many applications and has many faults. A DES modification known as *triple DES* (*3DES*) was later proposed, in which the original algorithm was applied three times to increase the security. But it was found to be quite slow. The preferred algorithm today is the Advanced Encryption Standard (AES). The brute force attack is the only known possible way to break the AES algorithm [4].

STATE OF THE ART IN LIGHTWEIGHT ENCRYPTION

Information security has become more crucial than ever [8], [9]. Cryptographic algorithms have two preeminent subtypes: 1) symmetric key cryptography or private-key encryption and 2) public-key encryption or asymmetric key cryptography [10], [11]. Symmetric key cryptography is primarily based on the message knowing and using the same secret key between the sender and receiver nodes for encryption and decryption, respectively [12]. Thus, symmetric key cryptography usually has difficulty providing efficient, secure key management, particularly in open systems with a large number of users [13], [14].

The proposed work is based on public key message authentication codes. Two different cryptography algorithms are NtruEncrypt and the Rabin scheme. The latter is based mainly on the factorization problem of large values and thus is quite similar to the security approach of the Rivest–Shamir–Adleman (RSA) algorithm with the same-size modulus. This scheme also has an asymmetric computational cost. However, the encryption computation is extremely fast, and the decryption process time is comparable to the RSA of the same modulus. The NtruEncrypt algorithm is a relatively new

cryptosystem that claims to be highly effective and efficient and is thus particularly suitable for embedded applications, such as radio-frequency identification tags and smart cards, while offering a decent level of security commensurate with that of other recognized schemes, such as RSA.

Various cryptographic algorithms have been compared in the past. In one study, the performance matrix was calculated with simulation text files varying from 20 to 99,000 kB being encrypted. To obtain the best performance from the proposed algorithm, the implementation was well tested and optimized in terms of programming. The performance metric was throughput, and the encryption and decryption process throughputs were calculated. For the encryption process, the throughput was found by dividing the average of the total plaintext in kilobytes by the average encryption time. In the case of the decryption process, throughput was calculated as the total ciphertext average divided by the average decryption time.

The performance evaluation of selected symmetric cryptographic algorithms was presented. The selected algorithms were Blowfish, AES, DES, and 3DES. First, the results of the presented simulation showed that Blowfish performed better than other encryption algorithms, trailed by AES in terms of throughput. Second, 3DES was the least efficient of them all. The study also examined the energy consumption of different but commonly used symmetric key protocols on handheld devices. It was found that after only 600 encryptions of a 5-MB file using the 3DES protocol, the remaining battery power was 45% and subsequent encryptions were not possible, as the battery died rapidly thereafter.

An analytical comparison of the RC4 and AES cryptographic algorithms has also been performed [3]. The results indicate that RC4 is energy efficient and with faster data encryption–decryption. On the basis of the analysis in this research document, RC4 was considered to be quite a bit more effective than AES. The RC4 and AES encryption times were compared on various data packet sizes, the RC4 time being notably less than that of AES. An executable program was made, which showed that Blowfish was superior in performance to other general-use encryption algorithms. As AES needs additional computing power, it performed poorly compared to the other. RSA is suitable for business applications because it has maximum security built in. For each of the algorithms, the computational running times were calculated with respect to their key generation and encryption–decryption mechanisms.

> LEAIoT basically enlists the previously noted advantages of both the symmetric and asymmetric key algorithms.

LEAIoT: PROPOSED ENCRYPTION ALGORITHM

The encryption and decryption architecture is shown in Figure 2. Encryption starts with receiving plaintext from the sender and a private key n. Then, the plaintext is given a synthetic value. In the next step, the initial ciphertext (encrypted text) is obtained by applying key n to the synthetic value. After this, a new linear block cipher (NLBC)—so named because it is based on linear algebra—is used that generates a

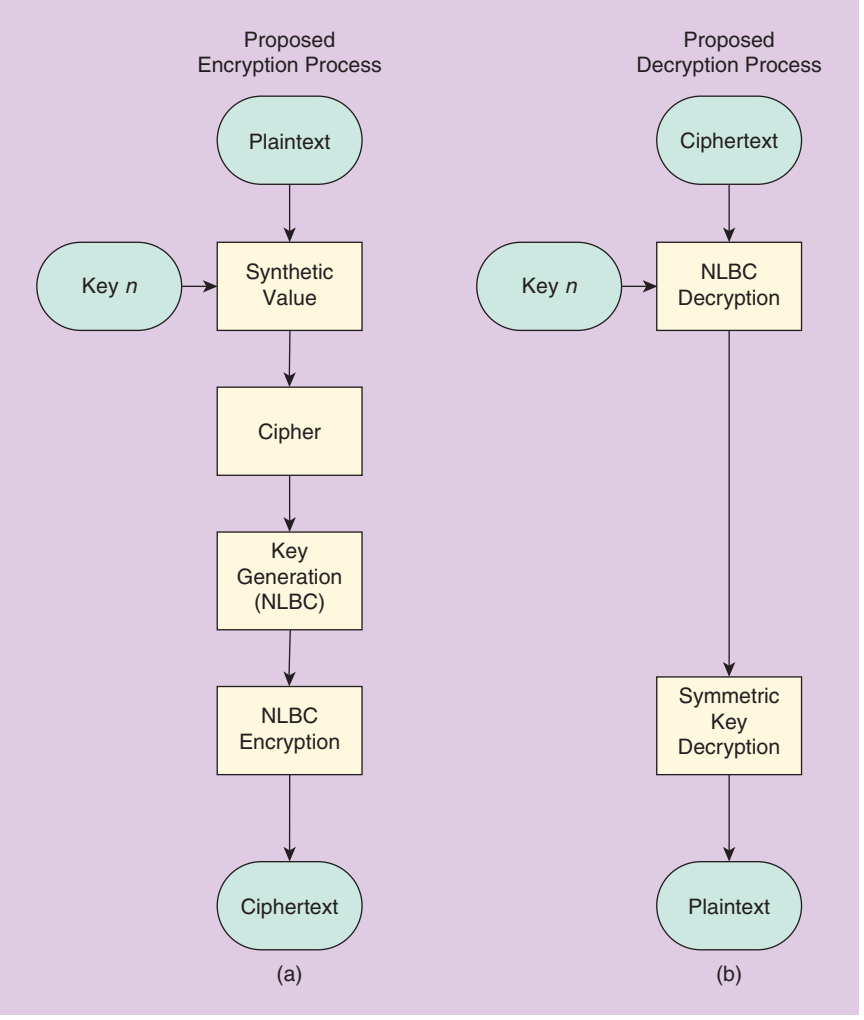

FIGURE 2. The LEAIoT proposed algorithm design: the (a) encryption design and (b) decryption design.

For the purpose of securing data in an efficient time span, LEAIoT implements symmetric encryption at the top of the encryption process.

private key. The NLBC private key is then applied to the ciphertext to achieve NLBC encryption. In the final step, a ciphertext is obtained and sent through the communication medium to the receiver. The receiver applies a decryption process to the obtained ciphertext, resulting in plaintext.

LEAIoT DESIGN

LEAIoT basically enlists the previously noted advantages of both the symmetric and asymmetric key algorithms. To encrypt the plaintext, LEAIoT first implements symmetric key encryption with a private key n that is known to both the sender and receiver. Then it implements the asymmetric linear block cipher technique with one key to make the encrypted text more secure by using one shared key of length n_1 and one NLBC-generated private key.

SYMMETRIC KEY DESIGN

For the purpose of securing data in an efficient time span, LEAIoT implements symmetric encryption at the top of the encryption process. LEAIoT lets the sender provide the shared private key n, because it is private but also shared by both the sender and receiver. Then the key n is modified by the system and generates a new, system-defined secure symmetric key (SSK). The plaintext is encrypted by applying the synthetic value and the SSK.

- *SSK generation*: the generation mechanism is quite simple and is performed in the following steps:
  - Get input n (private key) from the sender.
  - Calculate the inverse of n with the use of modulo 37.
  - The result of these steps is SSK k.
- *Symmetric encryption process*: the plaintext given by the sender is initially encrypted according to this process:
  - Assign synthetic values to the plaintext.
  - Calculate the ciphertext by multiplying the synthetic values of the plaintext by the synthetic value of n.
  - Find the ciphertext with modulo 37.
- *Symmetric decryption process*: in the decryption process, the following steps are performed:
  - Multiply the received ciphertext by SSK k.
  - Find the modulo 37 of the resultant value.
  - The result is the desired plaintext.

LINEAR BLOCK CIPHER

LEAIoT encrypts alphabets and numbers by using a new asymmetric key algorithm, the NLBC. Just like all asymmetric algorithms, it implements two keys, one open (public) and the other secret (private). With the help of both, the ciphertext or encrypted message is decrypted on the receiving side and vice versa.

- *NLBC encryption*: the developed NLBC algorithm performs these steps to encrypt the message:
  - Store the message that is to be encrypted in a string variable m.
  - Use the algorithm to select a $k \times k$ square and invertible matrix, also known as *private key* k_1.
  - Determine the n_1 integer value by calculating the shared key length.
  - Arrange the message m in a block form according to the selected square matrix.
  - Get a product of the arranged block of the message with the integer value n_1 and a square matrix.
  - Calculate modulo 37 with the result of the previous step. This produces the encrypted text or ciphertext.
  - Finally, direct the ciphertext toward the receiving side via a secure medium.
- *NLBC decryption process*: in the NLBC decryption process, the received ciphers are decoded with the help of the received keys and some other calculations. The sequence of the decoding process of our NLBC algorithm is as follows:
  - Receive the encrypted text.
  - Organize the ciphertext as r blocks.
  - Process the private key k_1 matrix with ciphertext and n_1.
  - Carry out the results of the previous steps with modulo 37.
  - The residue value should be the original message or plaintext.

LEAIoT IMPLEMENTATION

For the LEAIoT implementation, we selected the sample text *MSOFFICE2013* for encryption and decryption.

ENCRYPTION OF PLAINTEXT

In encryption, the first step is key generation, which is performed as follows:

- The user private key is n (e.g., $n = 12{,}345$).
- The modular inverse with 37 of $n = 12{,}345$ is 17 (proof: $1234517 \bmod 37 = 1$).
- Hence, SSK $n - 1 = 17$.

Now we have the SSK, i.e., $n - 1$, so the next phase of our symmetric key algorithm is encrypting the plaintext with n, as shown in Table 1.

Hence, the ciphertext obtained by applying n is PL06648I3SFQ, which is further processed with NLBC to make it more secure from and unreadable by unauthenticated sources. The NLBC algorithm for the encryption of the message is processed according to the following sequence:

- The message m is $m =$ PL06648I3SFQ.
- Let us create a 3×3 matrix as

$$k_1 = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \text{ and } k_1' = \begin{bmatrix} 18 & 23 & 32 \\ 1 & 25 & 12 \\ 18 & 1 & 18 \end{bmatrix}.$$

- The length of the shared key n is $n_1 = 5$.
- The segregation of message m into blocks according to k_1 is shown in Table 2.
- Then, there is multiplication of k_1 and n_1 with blocks b_i (CT: ciphertext),

$$\begin{aligned} CT &= (k_1^* n_1^* b_1) \bmod 37 \\ &= \begin{bmatrix} 1 & 2 & 1 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * 5 * \begin{bmatrix} 16 \\ 12 \\ 27 \end{bmatrix} \bmod 37 \\ &= \begin{bmatrix} 335 \\ 1{,}430 \\ 2{,}255 \end{bmatrix} \bmod 37 = \begin{bmatrix} 2 \\ 24 \\ 35 \end{bmatrix}. \end{aligned}$$

For the remaining blocks, the same process is carried out.

$$CT = (k_1^* n_1^* b_2) \bmod 37 = \begin{bmatrix} 1 & 2 & 1 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * 5 * \begin{bmatrix} 33 \\ 33 \\ 31 \end{bmatrix} \bmod 37$$

$$= \begin{bmatrix} 650 \\ 2{,}415 \\ 3{,}870 \end{bmatrix} \bmod 37 = \begin{bmatrix} 21 \\ 10 \\ 22 \end{bmatrix}$$

$$CT = (k_1^* n_1^* b_3) \bmod 37 = \begin{bmatrix} 1 & 2 & 1 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * 5 * \begin{bmatrix} 35 \\ 9 \\ 30 \end{bmatrix} \bmod 37$$

$$= \begin{bmatrix} 415 \\ 1{,}825 \\ 2{,}935 \end{bmatrix} \bmod 37 = \begin{bmatrix} 8 \\ 12 \\ 12 \end{bmatrix}$$

$$CT = (k_1^* n_1^* b_4) \bmod 37 = \begin{bmatrix} 1 & 2 & 1 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * 5 * \begin{bmatrix} 19 \\ 6 \\ 17 \end{bmatrix} \bmod 37$$

$$= \begin{bmatrix} 240 \\ 1{,}040 \\ 1{,}670 \end{bmatrix} \bmod 37 = \begin{bmatrix} 18 \\ 4 \\ 5 \end{bmatrix}.$$

- Hence, the final ciphertext is BX8UJVHLLRDE.

DECRYPTION OF CIPHERTEXT

The decryption process begins with the NLBC decryption, and then decryption is performed with the SSK. As decryption takes place on the receiver side, the receiver performs the following steps to decode the ciphertext.

- The received ciphertext is CT = BX8UJVHLLRD, with a cipher value of 22425211022812121845, a public key of $n_1 = 5$, and also

$$k_1' = \begin{bmatrix} 18 & 23 & 32 \\ 1 & 25 & 12 \\ 18 & 1 & 18 \end{bmatrix}.$$

- Calculate the modular inverse $n_1 - 1$ with modulo 37: $n_1 - 1 = 15$.
- Decode the ciphertext as $(k_1' * \text{blocks} * n_1 - 1) \bmod 37$.

$$(k_1' * b_1 * n_1^{-1}) \bmod 37 = \begin{bmatrix} 18 & 23 & 32 \\ 1 & 25 & 12 \\ 18 & 1 & 18 \end{bmatrix} * \begin{bmatrix} 2 \\ 24 \\ 35 \end{bmatrix} * 15 \bmod 37$$

LEAIoT encrypts alphabets and numbers by using a new asymmetric key algorithm, the NLBC.

Table 1. The symmetric key encryption.

Plaintext	Synthetic Value (M)	Ciphertext	Ciphertext
M	13	16	P
S	9	12	L
O	15	27	0
F	6	33	6
F	6	33	6
I	9	31	4
C	3	35	8
E	5	9	I
2	29	30	3
0	27	19	S
1	28	6	F
3	30	17	Q

Table 2. The asymmetric encryption block arrangement.

Blocks	Ciphertext	Synthetic Value
Block 1	P L 0	16, 12, 27
Block 2	6 6 4	33, 33, 31
Block 3	8 I 3	35, 9, 30
Block 4	S F Q	19, 6, 17

$$(k1' * b_1 * n1^{-1}) \bmod 37 = \begin{bmatrix} 25{,}620 \\ 15{,}330 \\ 10{,}350 \end{bmatrix} \bmod 37 = \begin{bmatrix} 16 \\ 12 \\ 27 \end{bmatrix}.$$

Similarly, for the remaining blocks,

$$(k1' * b_2 * n1^{-1}) \bmod 37 = \begin{bmatrix} 18 & 23 & 32 \\ 1 & 25 & 12 \\ 18 & 1 & 18 \end{bmatrix} * \begin{bmatrix} 21 \\ 10 \\ 22 \end{bmatrix} * 15 \bmod 37$$

$$= \begin{bmatrix} 19{,}680 \\ 8{,}025 \\ 11{,}760 \end{bmatrix} \bmod 37 = \begin{bmatrix} 33 \\ 33 \\ 31 \end{bmatrix}$$

$$(k1' * b_3 * n1^{-1}) \bmod 37 = \begin{bmatrix} 18 & 23 & 32 \\ 1 & 25 & 12 \\ 18 & 1 & 18 \end{bmatrix} * \begin{bmatrix} 8 \\ 12 \\ 12 \end{bmatrix} * 15 \bmod 37$$
$$= \begin{bmatrix} 12,060 \\ 6,780 \\ 5,580 \end{bmatrix} \bmod 37 = \begin{bmatrix} 35 \\ 9 \\ 30 \end{bmatrix}$$
$$(k1' * b_4 * n1^{-1}) \bmod 37 = \begin{bmatrix} 18 & 23 & 32 \\ 1 & 25 & 12 \\ 18 & 1 & 18 \end{bmatrix} * \begin{bmatrix} 18 \\ 4 \\ 5 \end{bmatrix} * 15 \bmod 37$$
$$= \begin{bmatrix} 8,640 \\ 2,670 \\ 6,270 \end{bmatrix} \bmod 37 = \begin{bmatrix} 19 \\ 6 \\ 17 \end{bmatrix}.$$

- Hence, the cipher values are 1612273333313593019617 and the ciphertext is PL06648I3SFQ.
- Now, after NLBC decryption, the final phase is performed by symmetric decryption. It uses the shared private key to decrypt the NLBC resultant ciphertext. The SSK is used in this process. The sequence of the steps for symmetric decryption is the following.

Table 3. The symmetric decryption.

Ciphertext	Synthetic Value	Plaintext	Plaintext
P	16	13	M
L	12	19	S
0	27	15	O
6	33	6	F
6	33	6	F
4	31	9	I
8	35	3	C
I	9	5	E
3	30	29	2
S	19	27	0
F	6	28	1
Q	17	30	3

Table 4. The key generation times.

Key Size (bits)	RSA-AES (ms)	RSA-DES (ms)	Developed Algorithm (ms)
32	8	6	4
64	15	14	10
128	19	22	16
256	23	30	22

- There is a multiplication of the received ciphertext with the SSK $n-1$ and a calculation of mod 37, as shown in Table 3.
- Finally, we have our original text, i.e., MSOFFICE2013.

ANALYSIS OF THE CASE STUDY

LEAIoT was tested for its correctness, completeness, and efficiency (in terms of computational time). The results for these algorithms (for LEAIoT, this time includes both the SSK generation and NLBC key generation time) are recorded in Table 4. There, we also checked various key sizes against the other encryption algorithms and LEAIoT. The latter outperformed the other state-of-the-art algorithms in terms of key generation time, as shown in Figure 3.

The time taken by the algorithm in encryption and decryption is also quite important, as it is necessary for any encryption algorithm to encrypt the plaintext to ciphertext and then to decrypt the ciphertext to plaintext in as little time as possible. Thus, we checked the encryption–decryption process times of RSA-AES, RSA-DES, and LEAIoT. The encryption process time is equal to that of the decryption. Therefore, the results in Table 5 show the encryption–decryption process execution time.

In terms of various numbers of input plaintext bits and sizes, LEAIoT outperformed the existing algorithms in terms of process execution time as shown in Figure 4. Although the difference is small, in the real-time environment, every millisecond matters. LEAIoT is an effective lightweight algorithm because the key generation time and the time for encryption and decryption were calculated as the lowest in comparison with the state-of-the-art encryption algorithms. This provides low latency, especially in IoT communication scenarios.

CONCLUSION AND FUTURE WORK

A shared private key initially secures data by encrypting it with an SSK and decrypting it with the same key as used in LEAIoT. Afterward, the use of an asymmetric key with a linear block cipher algorithm assures that the data are secure. By using a block cipher algorithm in combination with a

FIGURE 3. The key generation time analysis.

Table 5. The data encryption–decryption time.

Number of Kilobits	RSA-DES (ms)	RSA-AES (ms)	Developed Algorithm (ms)
100	220	245	210
300	280	295	275
500	310	320	290
1,000	390	400	386

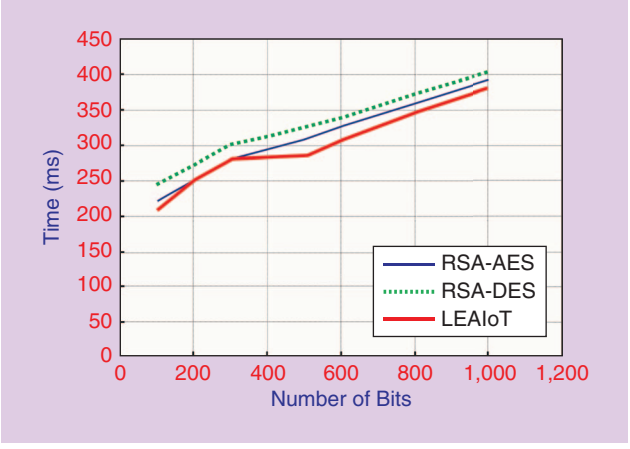

FIGURE 4. The data encryption–decryption time analysis.

symmetric algorithm, efficient results are produced. The linear block cipher results suggested the hypothesis that the use of RSA for the key management process in AES would produce an efficient algorithm. During the data transmission application, LEAIoT proved to be more effective than RSA-AES. In the future, researchers could use different combinations of various linear block ciphers with other symmetric key algorithms.

ACKNOWLEDGMENTS

This work was partially supported by national funding from the Fundação para a Ciência e a Tecnologia through the UID/EEA/500008/2013 Project; the government of the Russian Federation, grant 08-08, with resources from Funttel, grant 01.14.0231.00, under the Centro de Referência em Radiocomunicações project of the Instituto Nacional de Telecomunicações, Brazil; and by Brazilian National Council for Research and Development (CNPq) via grant 309335/2017–5.

ABOUT THE AUTHORS

Muhammad Asif Habib (drasif@ntu.edu.pk) is an assistant professor in the Computer Science Department, National Textile University, Pakistan.

Mudassar Ahmad (mudassar@ntu.edu.pk) is an expert in Linux networks.

Sohail Jabbar (sjabbar.research@gmail.com) is an assistant professor at the National Textile University, Pakistan.

Syed Hassan Ahmed (sh.ahmed@ieee.org) is an assistant professor in the Department of Computer Science, Georgia Southern University, Statesboro.

Joel J.P.C. Rodrigues (joeljr@ieee.org) is with the National Institute of Telecommunications, Brazil; Instituto de Telecomunicações, Lisbon, Portugal; ITMO University, Saint Petersburg, Russia; and University of Fortaleza, Brazil.

REFERENCES

[1] A. E. Mohammed and F. M. Abdalla. (2016). DES security enhancement using genetic algorithm. *SUST J. Eng. Comput. Sci. (JECS)*. [Online]. Available: http://jst.sustech.edu/content_details.php?id=1286&chk=1d0fe39d5a8005a3660b0a74d7957108

[2] C. Cheng, R. Lu, A. Petzoldt, and T. Takagi, "Securing the Internet of Things in a quantum world," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 116–120, 2017.

[3] A. A. Pammu, K. S. Chong, W. G. Ho, and B. H. Gwee, "Interceptive side channel attack on AES-128 wireless communications for IoT applications," in *Proc. IEEE Asia Pacific Conf. Circuits and Systems (APCCAS)*, 2016, pp. 650–653.

[4] P. Corcoran, "Mobile-edge computing and Internet of Things for consumers: Part II: Energy efficiency, connectivity, and economic development," *IEEE Consum. Electron. Mag.*, vol. 6, no. 1, pp. 51–52, 2017.

[5] M. Tausif, J. Ferzund, S. Jabbar, and R. Shahzadi, "Towards designing efficient lightweight ciphers for Internet of Things," *KSII Trans. Internet Inform. Syst.*, vol. 11, no. 8, 2017. doi: 10.3837/tiis.2017.08.014.

[6] R. Das and P. Chatterjee, "Securing data transfer in IoT employing an integrated approach of cryptography & steganography," in *Proc. Int. Conf. High Performance Compilation, Computing, and Communications (HP3C)*, 2017, pp. 17–22.

[7] Y. Nir, T. Kivinen, P. Wouters, and D. Migault, "Algorithm implementation requirements and usage guidance for the internet key exchange protocol version 2 (Ikev2)," Internet Eng. Task Force, Request for Comment (RFC) 8247, 2017. doi: 10.17487/RFC8247.

[8] J. Yang, S. He, Y. Lin, and Z. Lv, "Multimedia cloud transmission and storage system based on Internet of Things," *Multimedia Tools Appl.*, vol. 76, no. 17, pp. 17,735–17,750, 2017.

[9] K. J. Singh and D. S. Kapoor, "Create your own Internet of Things: A survey of IoT platforms," *IEEE Consum. Electron. Mag.*, vol. 6, no. 2, pp. 57–68, 2017.

[10] S. K. Yadav, "Some problems in symmetric and asymmetric cryptography," Ph.D. dissertation, Dept. Mathematics, Agra Univ., India, 2010.

[11] S. Shakil and V. Singh, "Security of personal data on Internet of Things using cryptographic algorithm," *Int. J. Eng. Sci.*, vol. 6, no. 4, pp. 3803–3805, 2016.

[12] M. Khan, S. Din, S. Jabbar, M. Gohar, H. Ghayvat, and S. C. Mukhopadhyay, "Context-aware low power intelligent smarthome based on Internet of Things," *Comput. Elect. Eng.*, vol. 52, pp. 208–222, May 2016.

[13] D. Park, "The quest for the quality of things: Can the Internet of Things deliver a promise of the quality of things?" *IEEE Consum. Electron. Mag.*, vol. 5, no. 2, pp. 35–37, 2016.

[14] P. Valerio, "Is the IoT a tech bubble for cities? With more cities joining the smart city revolution and investing in sensors and other IoT devices, the risk of a new tech bubble is rising," *IEEE Consum. Electron. Mag.*, vol. 5, no. 1, pp. 61–62, 2016.