Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.052124

ARTICLE

Time Parameter Based Low-Energy Data Encryption Method for Mobile Applications

Li-Woei Chen[1], Kun-Lin Tsai[2,*], Fang-Yie Leu[3], Wen-Cheng Jiang[2] and Shih-Ting Tseng[2]

[1]Department of Computer and Information Sciences, Chinese Military Academy, Kaohsiung, 830, Taiwan

[2]Department of Electrical Engineering, Tunghai University, Taichung, 407, Taiwan

[3]Department of Computer Science, Tunghai University, Taichung, 407, Taiwan

*Corresponding Author: Kun-Lin Tsai. Email: kltsai@thu.edu.tw

Received: 23 March 2024 Accepted: 29 May 2024 Published: 08 July 2024

ABSTRACT

Various mobile devices and applications are now used in daily life. These devices require high-speed data processing, low energy consumption, low communication latency, and secure data transmission, especially in 5G and 6G mobile networks. High-security cryptography guarantees that essential data can be transmitted securely; however, it increases energy consumption and reduces data processing speed. Therefore, this study proposes a low-energy data encryption (LEDE) algorithm based on the Advanced Encryption Standard (AES) for improving data transmission security and reducing the energy consumption of encryption in Internet-of-Things (IoT) devices. In the proposed LEDE algorithm, the system time parameter is employed to create a dynamic S-Box to replace the static S-Box of AES. Tests indicated that six-round LEDE encryption achieves the same security level as 10-round conventional AES encryption. This reduction in encryption time results in the LEDE algorithm having a 67.4% lower energy consumption and 43.9% shorter encryption time than conventional AES; thus, the proposed LEDE algorithm can improve the performance and the energy consumption of IoT edge devices.

KEYWORDS

Mobile application security; AES; data encryption; time parameter; mobile device

Nomenclature

T_S and T_R	System times of the sender S (user device) and receiver R (the devices in the access network and serving core network)
α, β, and γ	Interpretation 2 Random numbers generated by the user device
KA, KB, and KC	128 × 8-bit time arrays used to generate parameter arrays
FlagArr	Flag array for identifying the elements in the dynamic S-Box. The FlagArr ensures that each element in the dynamic S-Box is unique
PA1, PA2, and PA3	Parameter arrays derived from KA, KB, and KC, respectively, are employed to remove repeated elements and generate the dynamic S-Box

RPA	Residual parameter array comprising the unselected values of FA
RLi and RLR	Local variables used to record the length of PAi ($1 \leqq i \leqq 3$) and IRA
V_e and V'_e	Verification messages that are calculated by the sender and receiver by using a predefined hash function.
$LEDE_{old}$ and $LEDE_{new}$	Data encryption with the old and new dynamic S-Box
$LEDE_{old}^{-1}$ and $LEDE_{new}^{-1}$	Data decryption with the old and new dynamic S-Box
ΔT	Time difference, including the data processing time of the sender and receiver and a reasonable communication time
ACK	Acknowledgment message used to ensure that the sender and receiver have the same dynamic S-Box and inverse dynamic S-Box

1 Introduction

In the past decade, rapid improvements in communication technologies have enabled numerous applications of mobile communication, artificial intelligence, and wearable devices. Some of these applications require high-speed data processing [1], some require low energy consumption [2], and some require high data security [3]. In order to meet these disparate requirements, numerous mobile communication protocols have been developed [4,5]. For example, Long Range Wide Area Network (LoRaWAN) protocol [6,7] uses the Advanced Encryption Standard (AES) [8] and a low-power and long-range physical proprietary radio communication technique to enable data from numerous user equipment (UE) to be securely transmitted to the devices in the access network and serving core network with low power consumption.

In many mobile applications, UE must preprocess raw data and encrypt them before transmission. Many devices collect personal data for transmission to other mobile devices and cloud servers; thus, these data should be encrypted efficiently. For example, many wearable devices collect or track user vital signs or health data; because of the risk of data leakage, ensuring data safety is paramount [9].

Although numerous encryption algorithms can be utilized by mobile devices [10,11], AES is commonly used because of its security features; it ensures data integrity and confidentiality and can be employed for point-to-point encryption. However, many UEs rely on batteries and have limited computing capability. Although AES ensures data security, it is energy-intensive and can rapidly reduce the available device energy. Hence, reducing energy consumption while ensuring data security is crucial for extending the device's lifetime.

This study proposes a low-energy data encryption (LEDE) method in which a modified version of AES and device system time are employed to enhance the security of the transmitted data and reduce the energy consumption of mobile devices. In LEDE, a dynamic substitution table (S-Box) is derived from the system time and used instead of the static S-Box in the conventional AES encryption process. A new dynamic S-Box is generated periodically. This method reduces the number of rounds in the encryption and decryption processes without reducing security, enabling a balance between security and energy consumption. Simulation results indicate that the LEDE algorithm successfully reduces energy consumption and is suitable for modern mobile communication systems. The contributions of this paper are as follows:

(1) Designing a system-time-based dynamic S-Box for AES encryption that improves the security of mobile applications;

(2) Reducing the number of AES encryption rounds to reduce energy consumption;

(3) Presenting a secure method to transmit the system time and encrypted data between the sender and the receiver in a mobile communication system.

The rest of this paper is organized as follows: Section 2 introduces the AES encryption algorithm and reviews the relevant studies. Section 3 describes the proposed LEDE algorithm and the dynamic S-Box generation process. Section 4 presents the mathematical model of the LEDE. Section 5 details the energy and performance simulation and security analyses of the proposed scheme. Finally, in Section 6, the conclusions of this study and directions for future research are provided.

2 Preliminary

2.1 Overview of the AES Encryption Algorithm

AES [8], established by the US National Institute of Standards and Technology (NIST) in 2001, is a well-known data encryption method. AES adopts a symmetric block cipher scheme with an encryption-decryption key of length 128, 192, or 256 bits based on the security requirements. Depending on the key length, AES performs 10 to 14 encryption–decryption rounds, each of which, except the last round, comprises four steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The conventional 128-bit AES encryption process is depicted in Fig. 1.

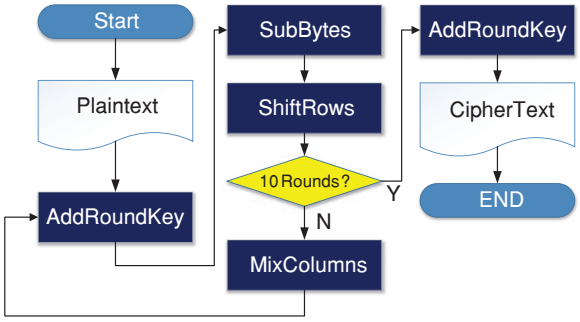

Figure 1: Conventional 128-bit AES encryption process

In AES, SubBytes is a nonlinear and invertible transformation step. In SubBytes, a substitution table, namely S-Box, is employed for individually mapping bytes of a data array to other bytes. S-Box entries are produced by calculating multiplicative inverses in a Galois Field utilizing an affine transformation. In ShiftRows, byte transposition is performed by rotating rows of the data array in accordance with predefined offsets. In MixColumns, each column of the data array is multiplied by a modular polynomial equation or prime number in a Galois field. Instead of directly performing this step, MixColumns can also be implemented using large lookup tables. In AddRoundKey, the key for the current round (the round key, which is derived from an initial encryption key in the key expansion unit) is added to the data array by performing an exclusive-OR (XOR) operation on each byte of the data block with the corresponding byte in the round key.

2.2 Related Studies

Security is critical in many mobile communications and Internet-of-Things (IoT) systems, such as medical or military communication systems. Numerous studies have investigated mobile communications and IoT security; some have focused on data confidentiality and integrity, while others have emphasized network security management. AES is widely used to encrypt mobile communications

and IoT data because of its simple encryption and decryption processes. Many studies on AES have proposed methods for improving its security or reducing its power consumption.

In the AES SubBytes step, a well-designed S-Box can considerably reduce the strength of the relationship between plaintext and its ciphertext [12,13] and improve the ciphertext's confusion, enhancing resistance to brute-force attacks. Therefore, some studies have proposed improvements to the structure of the S-Box in the SubBytes step [14–16]. For example, Liu et al. [17] designed an algorithm called GDBRK (Generation of D-Box and Round Keys) to generate several S-Boxes to replace AES to increase security. D'souza et al. [18] conducted XOR operations on round keys and replaced the S-Box elements. Their method improved the ciphertext confusion and effectively defended against attacks. Inspired by the butterfly effect in chaos theory, Talirongan et al. [19] utilized the high sensitivity of ciphertext to parameters and input values. They added a step to the AES algorithm to ensure that small changes in the input values resulted in substantially different ciphertext, thereby improving the overall diffusion and confusion. Tsai et al. [20] employed system time to improve security; the time difference between the sender and the receiver was utilized to resist replay attacks. The time difference includes the time required for data processing and transmission. Messages are accepted if the time difference is within an acceptable error range; otherwise, they are discarded. Singh et al. [21] constructed a dynamic S-Box using dynamic irreducible polynomials and an affine constant to improve the security of AES. For five analyses, they used grayscale and color images as the input plaintext images. However, all the studies above proposed successful S-Box modifications; a method for securely delivering new S-Boxes to the sender and receiver must still be developed. In addition, although frequently updating the S-Box improves attack resistance, S-Box generation requires substantial energy consumption.

Maitra et al. stated in [22] that security is crucial for mobile devices; however, edge node devices often have limited computing resources, such as memory and power. In order to reduce the energy consumption of mobile devices, some studies have focused on reducing end-device power consumption [23,24]; others have focused on the energy consumption of data encryption and communications [25,26]. Maitra et al. [22] performed a comparative study of AES with and without hardware accelerators. They executed an extended tiny encryption algorithm to analyze the accelerators' performance regarding memory usage, power consumption, and execution time. They concluded that hardware accelerators enhance AES's data encryption performance. Guo et al. [25] proposed an AND–rotation–XOR operation based on a logical combination method to overcome the shortcomings of existing lightweight encryption algorithms. They used a round-based and serial hardware architecture to implement their method and claimed that a round-based architecture has a higher throughput and consumes less energy than a serial hardware architecture. Kane et al. [26] analyzed the power consumption, time, and energy costs of three ciphers: AES, ChaCha, and Acorn. Acorn and ChaCha were substantially faster than AES and used less energy than AES; thus, they suggested that the Acorn and ChaCha algorithms should be considered for lightweight encryption applications. Manjula et al. [27] proposed a method to dynamically generate AES S-Boxes based on the round key for each encryption round. This dynamic generation aims to increase the complexity and resistance of the AES cipher to various attacks by altering the S-box in every round, depending on the key. This approach enhances security by introducing more variability and randomness into the encryption process and maintains the fundamental operational principles of AES. Zahid et al. [28] proposed a square polynomial transformation combined with affine transformations and a dynamic permutation process to generate robust, key-dependent S-boxes. Their approach enables the creation of a vast array of dynamic S-boxes through minor adjustments in the transformation and permutation parameters, thus enhancing cryptographic strength. Their results demonstrated that the proposed S-box design not only meets

but surpasses many existing standards, making it a promising option for securing modern encryption systems.

3 Dynamic S-Box and LEDE Method

This section introduces the dynamic S-Box generation method and then presents the proposed LEDE method.

3.1 Dynamic S-Box Generation Method

As described in Section 2, AES uses an S-Box in the SubBytes step of each round for individually substituting the bytes of a data array with other bytes. Because the S-Box is a fixed-value substitution box and plays a vital role in the AES encryption process, the S-Box structure should be optimized to improve AES security. In [16,29], a fixed string and three inner keys were employed to generate the dynamic S-Box. In contrast, this study employed the device system time to generate the dynamic S-Box to replace the conventional static S-Box. Using the device system time not only allows a further improvement in AES security, as demonstrated in the literature [29], but also enables the resistance of replay and eavesdropping attacks during data and parameter transmission.

Before data transmission, the IoT end device captures its system time to generate a dynamic S-Box. The system time is expanded into three arrays, KA, KB, and KC, used in the dynamic S-Box generation process. The array generation algorithm is presented in Algorithm 1. Three random numbers, namely α, β, and γ, for increasing the entropy of the time arrays are also generated. These three random numbers, α, β, and γ, are generated using a cryptographically secure pseudo-random number generator (CSPRNG) that meets the current standards for cryptographic applications. The CSPRNG is seeded with the device's system time and hardware-based entropy sources to ensure randomness. These random numbers are used in the parameter transmission process.

Algorithm 1: Generation of Time Arrays KA, KB, and KC

1: **Input:** System Time *Ts*
2: **Outputs:** Time arrays KA, KB, KC and Random numbers α, β, γ
3: Generate random numbers α, β, and γ
4: Put System Time *Ts* into KA;
5: Put System Time *Ts* into KB;
6: **for** ($i = 0$ to 63)
7: $\quad$ KB[i] = KB[i] + α;
8: $\quad$ KB[127−i] = KB[127−i] + β;
9: $\quad$ Swap KB[i] and KB[127−i];
10: $\quad$ KC = (KA ∗ KB + γ) mod 256;

The dynamic S-Box can be generated after the system-time-based time arrays are obtained. The dynamic S-Box generation algorithm is displayed in Algorithm 2. The inputs of this algorithm are three-time arrays KA, KB, and KC, which are generated using the algorithm shown in Algorithm 2. In Algorithm 2, a flag array (FlagArr) is employed to ensure that the elements in the dynamic S-Box are nonduplicate. Three parameter arrays, PA1, PA2, and PA3, are utilized to generate the contents of the dynamic S-Box, and one residual parameter array (RPA) is created to record the parameters that do not appear in PA1, PA2, or PA3. When all elements are in PA1, PA2, PA3, and RPA, step 20 of the dynamic S-Box generation algorithm is performed. The non-null elements of PA1, PA2, PA3, and RPA are sequentially integrated into the dynamic S-Box for use in data encryption.

Algorithm 2: Dynamic S-Box Generation

```
Input: Time arrays KA, KB, KC
Output: Dynamic S-Box
Initialize a 256-element flag array FlagArr and four parameter array PA1, PA2, PA3, and RPA
Initialize all parameters to 0, including RL1, RL2, RL3, RLR;
for (i = 0 to 127) // initialize the elements for PA1, PA2, and PA3, and
{   //   guarantee each value of PA1, PA2, and PA3 is unique by using FlagArr
  if (FlagArr[KA[i]] is null)   // check whether the value of KA[i] is being used or not
  { Copy KA[i] to PA1[RL1];   // initialize the value of PA1 by using time array KA
     Let FlagArr[KA[i]] = Ture;
     RL1 = RL1 + 1;}
  if (FlagArr[KB[i]] is null)   // check whether the value of KB[i] is being used or not
  { Copy KB[i] to PA2[RL2];   // initialize the value of PA1 by using time array KA
     Let FlagArr[KB[i]] = Ture;
     RL2 = RL2 + 1;}
  if (FlagArr[KC[i]] is null)   // check whether the value of KC[i] is being used or not
  { Copy KC[i] to PA3[RL3];   // initialize the value of PA1 by using time array KA
     Let FlagArr[KC[i]] = Ture;
     RL3 = RL3 + 1;}}
for (m = 0 to 255) //check the non-assigned value
{   if FlagArr[m] is null
  { RPA[RLR] = m;
     RLR = RLR + 1;}
Sequentially integrate nonnull elements of PA1, PA2, PA3, RPA into the dynamic S-Box;
```

The dynamic S-Box can be easily generated by UEs because no complex operations are required for the algorithms displayed in Algorithm 1 and Algorithm 2. In secure IoT communications, the sender uses a dynamic S-Box to encrypt data, whereas the receiver utilizes an inverse dynamic S-Box to decrypt data. Hence, the receiver must use a simple process to generate the inverse dynamic S-Box, and the UE must securely transmit the system time (T_S) and three random numbers, α, β, and γ, to the receiver (the devices in the access network and serving core network). The secure transmission method is described in Section 3.2.

The inverse dynamic S-Box required for the decryption process is generated by exchanging the values of each element in the dynamic S-Box matrix according to the element's row and column. Fig. 2 presents an example of transforming an inverse dynamic S-Box element. The value of the dynamic S-Box element is converted into the corresponding row and column values of the inverse dynamic S-Box element, and the row and column values of the dynamic S-Box element are transformed into the value of the inverse dynamic S-Box element. In Fig. 2, the value of an element is 37, and its row and column coordinates are (C, 9). Therefore, the element at (3, 7) in the inverse dynamic S-Box has a value of C9.

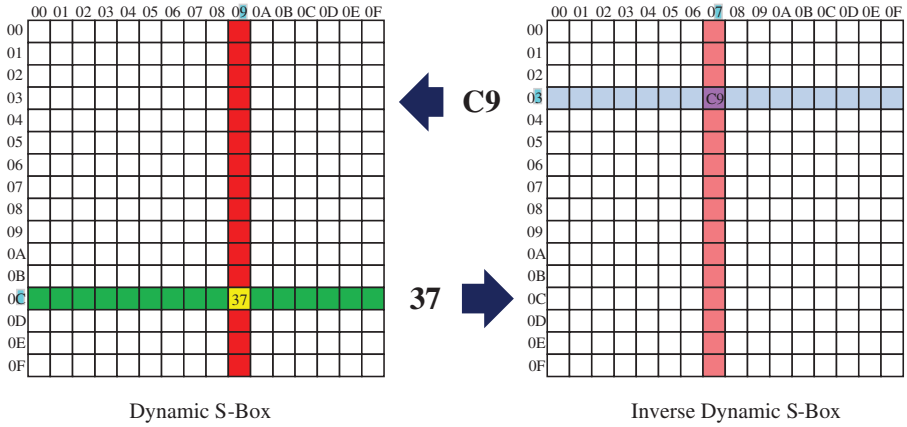

Figure 2: Inverse dynamic S-Box element transformation

Because the dynamic S-Box and inverse dynamic S-Box are created using the UE's system time, the dynamic S-Box update frequency strongly affects communication performance and security. Generating a new dynamic S-Box before every data transmission is inefficient. Hence, in the proposed LEDE method, a new dynamic S-Box and its corresponding inverse dynamic S-Box are generated periodically, such as once per day or week. The decision to update the dynamic S-Box daily or weekly is based on a balance between security enhancement and operational efficiency. Frequent updates improve security by reducing the vulnerability window when an attacker can exploit a known S-Box. However, too frequent updates could increase computational overhead and disrupt system operations. This study generates a new S-Box daily or weekly to optimize this trade-off based on typical data sensitivity and attack models pertinent to the deployment environments, such as mobile networks or IoT systems. This frequency can be adjusted based on specific security needs and system capabilities. Because the system time used in the LEDE method can be accurate to the millisecond and the UE generates three random numbers, an attacker cannot generate the same dynamic S-Box as the UE; even an error of 1 ms in the guessed time generates an incorrect dynamic S-Box. The new system time and random numbers delivered to the receiver are encrypted using AES with the previous dynamic S-Box. Once the receiver obtains the new system time from the sender, the receiver can generate the same dynamic S-Box and inverse dynamic S-Box as those generated by the sender for encryption and decryption, respectively. The transmission process for the new system time and three random numbers is described in the following section.

3.2 LEDE Method

The sender, i.e., UE, uses 128-bit AES to encrypt data before transmission, and the receiver, i.e., the device in the access network and serving core network, decrypts the messages. The sender and receiver must synchronize their system times before initiating the LEDE data encryption and decryption process to mitigate the system time synchronization error. The LEDE data encryption–decryption process is depicted in Fig. 3. After a predefined period, the UE updates the dynamic S-Box based on the algorithms described in the previous section. The system time and three random numbers are transmitted to the receiver using a secure parameter transmission process, as shown in Fig. 4. Alternatively, the data (i.e., plaintext) can be encrypted using AES with the existing dynamic S-Box. However, the number of encryption rounds in the LEDE method is less than those in conventional AES. Based on the experiments described in Section 4, the proposed LEDE method only performs six

rounds of encryption to achieve the same security level as 10-round AES. Therefore, in the AES-based LEDE data encryption–decryption process, k can be set as 6. If higher security than AES is desired, k can be set to a higher number; however, a trade-off exists between security and energy consumption during encryption and decryption.

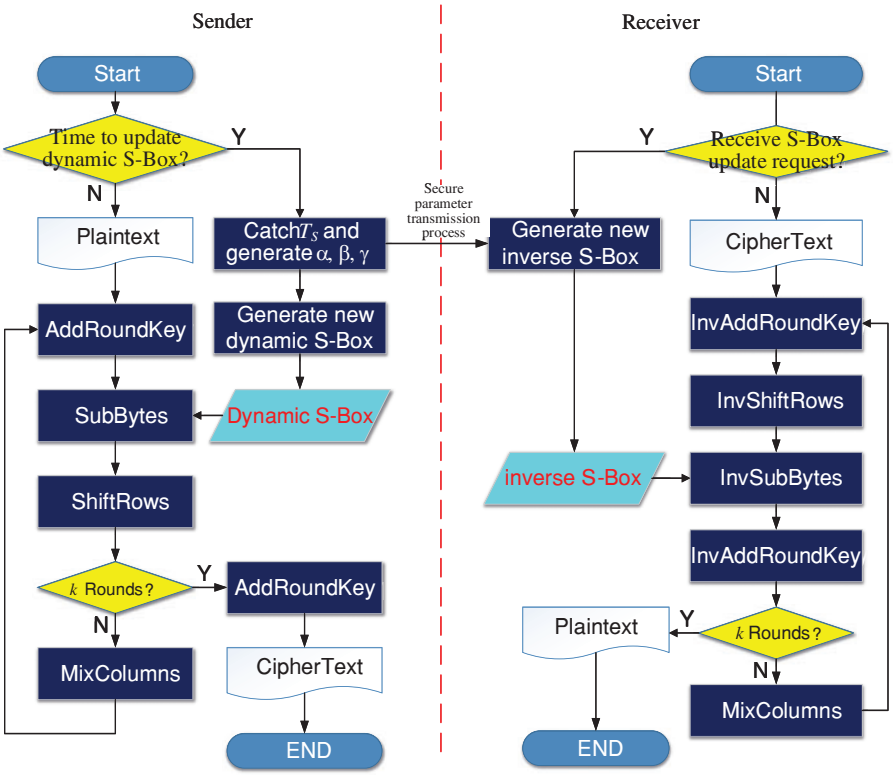

Figure 3: AES-based LEDE data encryption and decryption process

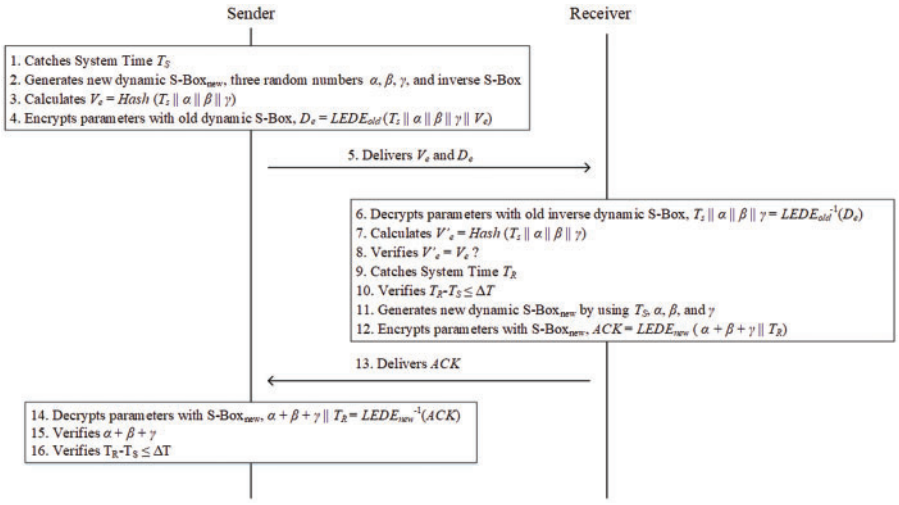

Figure 4: Secure parameter transmission process

For securely transmitting the system time and random numbers to the devices in the access network and serving core network, a secure parameter transmission process was designed (Fig. 4). Initially, the UE performs the following steps in sequence:

(1) The UE catches its system time T_S.

(2) It generates three random numbers, α, β, and γ; the new dynamic S-Box; and the new inverse dynamic S-Box using the algorithms displayed in Algorithm 1 and Algorithm 2.

(3) It calculates a hash value $V_e = Hash(T_S||\alpha||\beta||\gamma)$, where Hash indicates a hash function, and || concatenates different signals.

(4) It encrypts TS, α, β, γ, and Ve by using the AES-based LEDE method with the old dynamic S-Box; that is, $D_e = LEDE_{old}(T_S||\alpha||\beta||\gamma||V_e)$;

(5) It transmits V_e and D_e to the receiver.

When the devices in the access network and serving core network receive V_e and D_e, they perform the following steps in sequence:

(6) The receiver decrypts D_e by calculating $T_S||\alpha||\beta||\gamma||V_e = LEDE^{-1}_{old}(D_e)$ to obtain T_S, α, β, γ, and V_e.

(7) It then calculates $V_e' = Hash(T_S||\alpha||\beta||\gamma)$.

(8) It checks whether $V_e' = V_e$; if they are unequal, the parameters differ from those of the user device. The receiver then halts the S-Box update process and responds to the user device; otherwise, the receiver performs step 9.

(9) It catches the receiver's system time T_R.

(10) It verifies whether $T_R - T_S \leq \Delta T$, where ΔT is the expected difference between T_R and T_S, which includes the processing time for steps 2–9, a reasonable communication time for step 5, and time discrepancies due to synchronization errors. If $T_R - T_S > \Delta T$, the receiver halts the S-Box update process and responds to the UE; otherwise, it performs step 10.

(11) It generates a new dynamic S-Box and inverse dynamic S-Box using T_S, α, β, and γ.

(12) It encrypts $\alpha + \beta + \gamma$ and T_R using the new dynamic S-Box to generate the ACK message; that is, $ACK = LEDE_{new}(\alpha + \beta + \gamma||T_R)$.

(13) It delivers ACK to the UE.

When the UE receives the ACK message from the devices in the access network and serving core network, the UE performs the following steps in sequence:

(14) It uses the new dynamic S-Box to decrypt ACK by calculating $\alpha+\beta+\gamma||T_R = LEDE^{-1}_{new}(ACK)$.

(15) It uses α, β, and γ from its memory to calculate $\alpha + \beta + \gamma$ and then compares the calculated value with the received value. If these values are unequal, the UE halts the S-Box update process and responds to the receiver; otherwise, it performs step 16.

(16) It verifies whether $T_R - T_S \leq \Delta T$. If not, the UE halts the S-Box update process and responds to a receiver; otherwise, it completes the secure parameter transmission process.

When the UE and receiver have generated the new dynamic S-Box and corresponding inverse dynamic S-Box, the transmitted data can be encrypted and decrypted using the new dynamic S-Box. It is worth noting that each application pair (user end and application end) requires a unique data encryption key, leading to a scenario where a User Equipment (UE) holds multiple keys.

4 Mathematical Model of the LEDE

This section presents the developed mathematical model for the LEDE method based on dynamic S-Box generation and secure parameter transmission. The model will encapsulate system times, random number generation, parameter arrays, and the encryption/decryption processes. This model abstracts the complex processes described in this paper into a mathematical framework. It encapsulates the steps in generating dynamic S-Boxes relying on system time, performing secure AES encryption with these S-Boxes, and securely transmitting parameters necessary for encryption and decryption between sender and receiver devices.

(1) System Times and Random Numbers
- Let T_S and T_R represent the system times of the sender and receiver, respectively.
- Let α, β, and γ represent random numbers generated by the user device.

(2) Time Arrays and Flag Array
- Time arrays are denoted as KA, KB, and KC, each derived from T_S.
- The flag array, FA, is employed to ensure uniqueness in the dynamic S-Box.

(3) Dynamic S-Box Generation
- Given KA, KB, and KC, the dynamic S-Box, DS, and the inverse dynamic S-Box, DS^{-1}, are generated as follows:
  - Initialize FA with 256 elements.
  - For $i = 0$ to 127, if $FA[KA[i]]$ is null, copy $KA[i]$ to $PA1[RL1]$, set $FA[KA[i]]$, and increment RL1; similar operations for $KB[i]$ and $KC[i]$.
  - Integrate non-null elements of $PA1$, $PA2$, $PA3$, and RPA into DS.

(4) Encryption and Decryption
- Encryption with dynamic S-Box: $Ciphertext = E_{DS}(Plaintext)$
- Decryption with inverse dynamic S-Box: $Plaintext = E_{DS^{-1}}(Ciphertext)$

(5) Secure Parameter Transmission

To securely transmit system time and random numbers:
- Sender encrypts T_S, α, β, γ, and verification message Ve with the old dynamic S-Box: $(De = E_{DS_{old}}(T_S||\alpha||\beta||\gamma||V_e)$.
- Receiver decrypts De to obtain T_S, α, β, γ, and Ve, then checks the validity of Ve and the time difference $T_R - T_S \leq \Delta T$.

(6) ACK Message Generation and Verification
- The receiver generates an ACK message using the new dynamic S-Box and sends it back to the sender for final verification.

5 Energy Usage and Security Analyses of the LEDE Method

A small IoT environment was constructed to verify the feasibility of the proposed LEDE method. The adopted end device consists of three major components: Raspberry Pi Model 4B with 8 GB of RAM, an MCP3008 analog-to-digital converter, and a pulse sensor. In the constructed environment, the pulse sensor senses a human pulse, and the sensed data are encrypted using the LEDE method on Raspberry Pi. The encrypted data are then transmitted to a remote server, and the server decrypts the data to obtain the original pulse waveform. This study analyzed its security features (Section 5.1). Hence, the energy consumed by the LEDE method was measured (Section 5.2).

5.1 Security Analysis

NIST SP 800-22 [30], a statistical test suite for determining the randomness of encrypted data, was used in the experiments. The evaluation metrics of NIST SP 800-22 are frequency, approximate entropy, and linear complexity, and the relevant results obtained in this study are listed in Table 1, which lists the security feature of conventional AES, Manjula's method [27], Zahid's method [28], and the proposed LEDE. Table 1 indicates that frequency exhibits the proportion of zeros and ones for the entire sequence; approximate entropy is the frequency of all possible overlapping m-bit patterns across the entire sequence; and linear complexity is the length of a linear feedback shift register.

Table 1: Security feature comparison

	Method	Frequency	Approximate entropy	Linear complexity
10 rounds	AES	0.5918	**0.0669**	0.6370
	[27]	0.5923	0.1573	0.7732
	[28]	0.5842	0.2311	0.8837
	LEDE	**0.5833**	**0.2213**	**0.8102**
8 rounds	AES	0.5042	0.0291	0.5345
	[27]	0.5510	0.0835	0.6840
	[28]	0.5674	0.1466	0.7625
	LEDE	**0.5655**	**0.1424**	**0.7314**
7 rounds	AES	0.4399	0.0099	0.3875
	[27]	0.4598	0.0643	0.5763
	[28]	0.4848	0.0914	0.6395
	LEDE	**0.4833**	**0.1009**	**0.6440**
6 rounds	AES	0.2133	0.0012	0.1233
	[27]	0.3956	0.0323	0.3969
	[28]	0.4093	0.0692	0.5014
	LEDE	**0.4110**	**0.0721**	**0.5245**

For sufficiently chaotic tests, 10-round AES had an approximate entropy of 0.0669, marginally less than that of six-round LEDE, indicating that six-round LEDE and 10-round AES have approximately the same security. Therefore, six-round LEDE can encrypt the sensed data from UE with equivalent security to that provided by 10-round AES; therefore, the proposed LEDE method requires lower encryption energy usage and encryption time than AES. When compared to the literature [27,28], reference [28] demonstrates superior performance in 10-round data encryption. However, LEDE achieves the best results in 6-round data encryption. This shows that the 6-round LEDE surpasses the security of 10-round AED and exceeds the security of previous studies.

Table 2 lists the security and encryption times achieved with 10-round AES, 6-round LEDE, 7-round [27], and 6-round [28] in three cases. Table 2 indicates that Case 1 and Case 3 are text documents, whereas Case 2 is a grayscale image. For these cases, 6-round LEDE, 7-round [27], 6-round [28], and 10-round AES exhibit similar approximate entropy. However, the encryption time of the LEDE method is consistently considerably lower than that of the others because fewer rounds are performed in the LEDE method. On average, six-round LEDE was 43.9% faster than 10-round AES for the same security level. In addition, the 6-round [28] exhibits similar encryption times to LEDE.

Table 2: Security and encryption times achieved with 10-round (10-R) AES, the six-round (6-R) LEDE, 7-round (7-R) [27], and 6-round (6-R) [28]

	Case 1 (MS Word document, 72 KB)				Case 2 (Grayscale pulse image, 3.4 MB)				Case 3 (Text, 214 KB)			
	10-R AES	6-R LEDE	7-R [27]	6-R [28]	10-R AES	6-R LEDE	7-R [27]	6-R [28]	10-R AES	6-R LEDE	7-R [27]	6-R [28]
Approximate entropy	0.0669	0.0681	0.0657	0.0675	0.0635	0.0648	0.0644	0.0653	0.0669	0.0714	0.0652	0.0694
Encryption time (ms)	0.9304	0.5062	0.6263	0.5156	4.0210	2.2224	2.7011	2.1964	1.2126	0.7106	0.8776	0.7174

5.2 Energy Consumption

The AES and LEDE algorithms were implemented in the Verilog Hardware Description Language [31] and then simulated using the ModelSim [32] simulation tool to evaluate the energy consumption of the proposed LEDE method and AES. The power consumption simulation results are listed in Table 3. In Table 3, Core Static and Core Dynamic represent the static and dynamic power consumption of the computational core, respectively. Static and dynamic power consumption represents the power consumed when the computational core is idle and during data processing. I/O indicates the combined power consumption of the input pins, output pins, and input/output buffers. As presented in Table 3, the power consumption for 10, 8, 7, and 6 rounds of AES and LEDE were evaluated. At a comparable security level, six-round LEDE consumed $[(569.02 - 341.36)/569.02] \times 100\% = 40.0\%$ less power than did 10-round AES.

Table 3: Power consumption of AES and the LEDE method under different numbers of rounds (unit: mW)

	10 Rounds		8 Rounds		7 Rounds		6 Rounds	
	AES	LEDE	AES	LEDE	AES	LEDE	AES	LEDE
Core static	0.50	0.51	0.50	0.51	0.50	0.51	0.50	0.51
Core dynamic	519.15	519.12	425.17	423.93	373.29	373.05	308.12	304.97

(Continued)

Table 3 (continued)

	10 Rounds		8 Rounds		7 Rounds		6 Rounds	
	AES	LEDE	AES	LEDE	AES	LEDE	AES	LEDE
I/O	49.37	47.22	43.74	43.46	40.99	39.73	36.85	35.88
Total	**569.02**	**566.85**	**469.41**	**467.90**	**414.78**	**413.29**	**345.47**	**341.36**

Energy is a product of power and computational time. A comparison of the energy consumption of 10-round AES and six-round LEDE is depicted in Fig. 5. In Case 1, the LEDE method exhibits up to 67.4% lower energy consumption than AES. Therefore, it concluded that the proposed LEDE method is a more efficient and less energy-intensive encryption method for UEs than conventional AES cryptography.

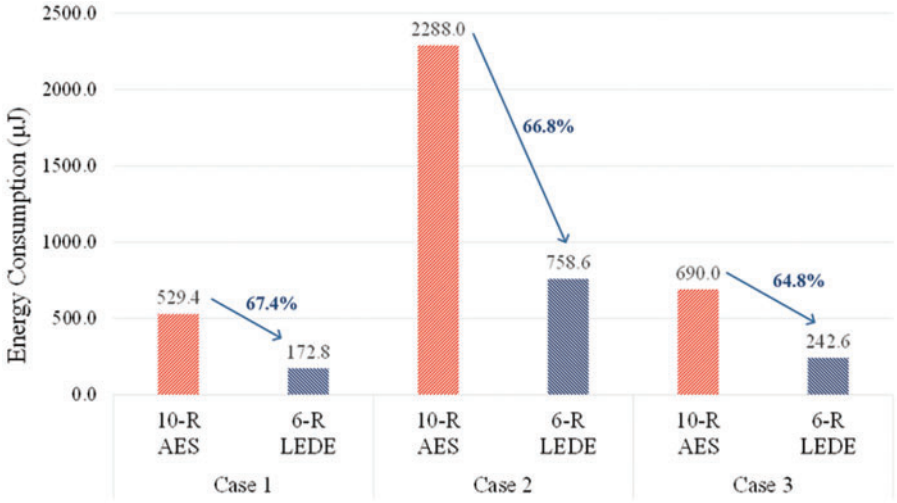

Figure 5: Energy consumption of AES and the LEDE method in three cases

6 Conclusion and Future Studies

Many UEs are powered by batteries and have limited computational resources. This study proposes a low-energy encryption method based on AES, LEDE, to improve data security and reduce the energy consumption of encryption. It uses a dynamic S-Box derived from the UE's system time in the LEDE encryption process. NIST SP 800-22 security analysis revealed that six-round LEDE and 10-round AES provide similar data security. Hence, the LEDE method requires less encryption time to achieve the same security as AES. If a new dynamic S-Box is generated once per day, the LEDE method can resist a known-key attack. In addition, the energy consumption of the LEDE method is up to 67.4% less than that of conventional AES. It concludes that the LEDE method suits energy-limited UEs that require high data security.

In the near future, the authors plan to extend this research to include detailed case studies on applying the LEDE algorithm in various sectors, such as smart home systems, healthcare monitoring, and industrial automation. These case studies will illustrate the algorithm's effectiveness in real-time

environments, focusing on specific challenges and benefits in each context. In addition, since a UE holds multiple keys, this necessity can impact data throughput when multiple applications on a UE must encrypt data simultaneously. This study aims to bridge this gap by incorporating scalability testing in future work. It intends to explore this issue further by conducting experiments to evaluate the robustness of the LEDE algorithm under various conditions of time synchronization discrepancies between devices. This will include a detailed analysis of how these errors could affect the encryption process's security integrity and operational efficiency and propose potential mechanisms to mitigate such risks.

Acknowledgement: The authors would like to thank all colleagues and students who contributed to this study. We are grateful to T. C. Liu and S. T. Lin for their collaboration during preliminary investigations. We thank the editor-in-chief and guest editors for constructive suggestions for an earlier version of this paper.

Funding Statement: This work was supported by the National Science and Technology Council, Taiwan, under Project NSTC 112-2221-E-029-015.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Li-Woei Chen, Kun-Lin Tsai, Fang-Yie Leu; algorithm design: Kun-Lin Tsai, Fang-Yie Leu; data collection: Wen-Cheng Jiang, Shih-Ting Tseng; experiment and simulation: Wen-Cheng Jiang, Shih-Ting Tseng; analysis and interpretation of results: Li-Woei Chen, Kun-Lin Tsai; draft manuscript preparation: Li-Woei Chen, Kun-Lin Tsai, Fang-Yie Leu. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: A Statistical Test Suite for Random and Pseudo-Random Number Generators for Cryptographic Applications. https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic (accessed on 15/04/2024).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Wang X, Yang LT, Song L, Wang H, Ren L, Deen MJ. A tensor-based multiattributes visual feature recognition method for industrial intelligence. IEEE Trans Ind Inform. 2020;17(3):2231–41. doi:10.1109/TII.2020.2999901.
2. Maitra S, Yanambaka VP, Puthal D, Abdelgawad A, Yelamarthi K. Integration of internet of things and blockchain toward portability and low-energy consumption. Trans Emerg Telecomm Technol. 2021;32(6):e4103. doi:10.1002/ett.4103.
3. Hatzivasilis G, Soultatos O, Ioannidis S, Verikoukis C, Demetriou G, Tsatsoulis C. Review of security and privacy for the internet of medical things (IoMT). In: Proceedings of 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS); 2019; Santorini Island, Greece. p. 457–64.
4. Sakamoto K, Liu F, Nakano Y, Kiyomoto S, Isobe T. Rocca: an efficient AES-based encryption scheme for beyond 5G. IACR Transacti Symmetr Cryptol. 2021;2021(2):1–30.
5. Porambage P, Gür G, Osorio DPM, Liyanage M, Gurtov A, Ylianttila M. The roadmap to 6G security and privacy. IEEE Open J Commun Soc. 2021;2:1094–122. doi:10.1109/OJCOMS.2021.3078081.

6. LoRa Alliance Technical Committee. LoRaWAN backend interfaces 1.0 specification. LoRa; 2017 Oct. Available from: https://resources.lora-alliance.org/technical-specifications/ts001-1-0-4-lorawan-l2-1-0-4-specification. [Accessed 2024].
7. LoRa Alliance Technical Committee. LoRaWAN 1.1 specification. LoRa alliance; 2017 Oct. Available from: https://resources.lora-alliance.org/technical-specifications/ts002-1-1-0-lorawan-backend-interfaces. [Accessed 2024].
8. Announcing the Advanced Encryption Standard (AES). Standard. Federal Information Processing, United States National Institute of Standards and Technology; 2001. Available from: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf. [Accessed 2024].
9. Arias O, Wurm J, Hoang K, Jin Y. Privacy and security in internet of things and wearable devices. IEEE Trans Multiscale Comput Syst. 2015;1(2):99–109. doi:10.1109/TMSCS.2015.2498605.
10. Sonya A, Kavitha G. A data integrity and security approach for health care data in cloud environment. J Internet Service Inform Secur. 2022;12(4):246–56. doi:10.58346/JISIS.2022.I4.018.
11. Nikitina V, Sánchez-Ancajima RA, Torres Rubio MA, Campos-Ugaz WA, Mejía Benavides A, Hende-Santolaya MR, et al. Enhancing security in mobile Ad Hoc Networks: enhanced particle swarm optimization-driven intrusion detection and secure routing algorithm. JoWUA. 2023;14(3):77–88.
12. Liloja RP. An intrusion detection system using a machine learning approach in IOT-based smart cities. J Internet Service Inform Secur. 2023;13(1):11–21. doi:10.58346/JISIS.2023.I1.002.
13. Hussain I, Shah T, Gondal MA, Khan WA, Mahmood H. A group theoretic approach to construct cryptographically strong substitution boxes. Neural Comput Appl. 2013;23(1):97–104. doi:10.1007/s00521-012-0914-5.
14. Shah T, Hussain I, Gondal MA, Mahmood H. Statistical analysis of S-box in image encryption applications based on majority logic criterion. Int J Phys Sci. 2011;6(16):4110–27.
15. Nitaj A, Susilo W, Tonien J. A new improved AES S-Box with enhanced properties. In: Information Security and Privacy: 25th Australasian Conference, ACISP 2020; Nov 30–Dec 2, 2020; Perth, WA, Australia: Springer International Publishing. p. 125–41.
16. Prasetiyo B, Ardian MN. Enhancement security AES algorithm using a modification of transformation ShiftRows and dynamic S-Box. J Phys: Conf Series. 2020;1567(3):32025.
17. Liu JJ, Huang YL, Leu FY, Pan XY, Chen LR. Generating dynamic box by using an input string. In: Proceedings of International Symposium on Mobile Internet Security; 2017; Singapore: Springer. p. 17–29.
18. D'souza FJ, Panchal D. Advanced encryption standard (AES) security enhancement using hybrid approach. In: Proceedings of 2017 International Conference on Computing, Communication and Automation (ICCCA); 2017; Greater Noida, India. p. 647–52.
19. Talirongan H, Sison AM, Medina RP. Modified advanced encryption standard using butterfly effect. In: Proceedings of 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM); 2018; Baguio City, Philippines. p. 1–6.
20. Tsai KL, Leu FY, Hung LL, Ko CY. Secure session key generation method for LoRaWAN servers. IEEE Access. 2020;8:54631–40.
21. Singh A, Agarwal P, Chand M. Image encryption and analysis using dynamic AES. In: Proceedings of International Conference on Optimization and Applications (ICOA); 2019; Kenitra, Morocco. p. 1–6.
22. Maitra S, Richards D, Abdelgawad A, Yelamarthi K. Performance evaluation of IoT encryption algorithms: memory, timing, and energy. In: 2019 IEEE Sensors Applications Symposium (SAS); 2019 Mar; Sophia Antipolis, France: IEEE. p. 1–6.
23. Filho RMPT, Oliveira LP, Carneiro LN. Security, power consumption and simulations in IoT device networks: a systematic review. In: International Conference on Advanced Information Networking and Applications; 2022 Mar; Cham: Springer International Publishing. p. 370–9.

24. Rosabal OM, López OLA, Pérez DE, Shehab M, Hilleshein H, Alves H. Minimization of the worst case average energy consumption in UAV-assisted IoT networks. IEEE Internet Things J. 2022;9(17):15827–38.
25. Guo Y, Li L, Liu B. Shadow: a lightweight block cipher for IoT nodes. IEEE Internet Things J. 2021;8(16):13014–23.
26. Kane LE, Chen JJ, Thomas R, Liu V, Mckague M. Security and performance in IoT: a balancing act. IEEE Access. 2020;8:121969–86.
27. Manjula G, Mohan HS. Constructing key dependent dynamic S-Box for AES block cipher system. In: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT); 2016 Jul; Karnataka, India: IEEE. p. 613–7.
28. Zahid AH, Rashid H, Shaban MMU, Ahmad S, Ahmed E, Amjad MT, et al. Dynamic S-Box design using a novel square polynomial transformation and permutation. IEEE Access. 2021;9:82390–401. doi:10.1109/ACCESS.2021.3086717.
29. Tsai KL, Huang YL, Leu FY, You I, Huang YL, Tsai CH. AES-128 based secure low power communication for LoRaWAN IoT environments. IEEE Access. 2018;6:45325–34. doi:10.1109/ACCESS.2018.2852563.
30. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, et al. A statistical test suite for random and pseudo-random number generators for cryptographic applications. Gaithersburg, MD, USA: NIST Special Publication; 2001. p. 800–22.
31. IEEE P1364-2005: IEEE standard verilog hardware description language; 2008. Available from: Verilog.com
32. ModelSim. 2022 Oct 18. Available from: https://www.intel.com/content/www/us/en/software-kit/750563/modelsim-intel-fpgas-pro-edition-software-version-21-1.html?wapkw=ModelSim [Accessed 2024].