



A novel energy-efficient encryption algorithm for secure data in WSNs

Haythem Hayouni¹ · Mohamed Hamdi¹

Accepted: 15 October 2020 / Published online: 26 October 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Wireless sensor networks (WSNs) are typically deployed environments, often very hostile and without assistance. A certain level of security must be provided. However, the resource constraint is the most important characteristic of this network. Indeed, WSNs are limited in terms of calculation, CPU, battery, etc. Therefore, a solution that aims to conserve these resources is widely desired. In this paper, we focus on the problem of encryption algorithm complexity. Some of the existing encryption algorithms provide high calculation cost, require large memory and are not applicable for WSNs scenarios. Based on Feistel structure, we propose a novel ultra-lightweight encryption algorithm, named ULEA. Besides diffusion and confusion of data, ULEA assumes minor encryption rounds with simplified transformations and functions to complex the cipher. The security analysis and experimental results suggest that the ULEA algorithm is an appropriate, low storage space, energy-efficient encryption process with high security for WSNs.

Keywords WSNs · Encryption algorithm · Feistel structure · Symmetric encryption

1 Introduction

Wireless sensor network (WSN) [20] is a set of autonomous sensors, equipped with computing capacity, storage, power and wireless transmission, which work together to observe a phenomenon in a physical environment. The sensor nodes have generally low computing, memory and energy, access to the radio medium is the single most expensive.

Wireless sensor networks are increasingly used in large systems monitoring applications in a variety of areas: the military, the environment, health, housing, etc. Sensor nodes that operate in hard to reach places, unprotected and battery recharging ability, can be subjected to many attacks. However, WSNs are

✉ Haythem Hayouni
haythem.hayouni@supcom.tn

¹ SupCom, University of Carthage, Carthage, Tunisia

typically deployed in hostile environments, and so a certain security level must be provided. On the other hand, since the sensors have limited computing resources, secure data in WSNs is a challenge and the limitations of sensors must be taken into consideration. The big problem is how to implement security techniques that require significant computing resources in technology with very limited resources.

The major security requirements [24] in WSNs are as follows:

- **Authentication:** it ensures that the data is actually transmitted by the supposed source.
- **Confidentiality:** it ensures that data is never disclosed to unauthorized nodes. This means that the information provided by each sensor node is accessible only by the node to which that information is intended.
- **Integrity:** it ensures that the received data is not altered during transit in network. In WSNs, where communication is often multi-hop, a compromise intermediate node can modify a message between transmitter and receiver.
- **Availability:** if this property is checked, the network will be able to ensure the availability of resources and the measured data, even with a few compromise nodes.

Cryptography [3] is the study of mathematical techniques that ensure some security services. It is defined as a science of converting information “plaintext” in encrypted data “ciphertext,” then from these encrypted data, to restore the original information. Cryptography is carried out according to certain tools. In cryptography, the ability to maintain a secret encrypted data, not based on algorithms, but on one secret information called key which is a parameter used as input of a cryptographic operation and must be used with the algorithms to produce the encrypted data. Encryption is the cryptographic system ensuring confidentiality, by using different keys. According to this use, there are two classes of encryption: symmetric encryption and asymmetric encryption.

Symmetric algorithms [21] are classified into stream ciphers and block. In block, encryption is done by a group of bits and bit-by-bit in stream cipher. The block encryption algorithm is better than stream cipher algorithm at computation levels. Most of the symmetric schemes use Feistel structure, in which the message is decomposed into two parts. These two parts undergo transformation functions over a determined number of rounds. The Feistel cipher presents an efficient technical to implement block symmetric encryption algorithms.

Based on the Feistel structure, this paper proposes a novel ultra lightweight encryption algorithm, called ULEA, to secure the transmitted data from sensor nodes to the base station. It assumes minor encryption rounds, simplified transformations and functions to provide an efficient security with a low resource consumption.

The major contributions of this paper as follows:

- We propose a novel resilient structure of the main encryption key used in our algorithm. So, if a sensor node is captured by an attacker, this key and the different generated subkeys (used for the different rounds) are still secure and cannot

be compromised. For this, it must be hard for an attacker to extract the encryption key concealed in our algorithm from only part of its implementation.

- In each round during the encryption process, efficient functions and operations are used to provide complete security for cipher against attacks. If the sensor node is captured, the attacker will not know the transmitted plaintext.
- We provide a security analysis and performance of the proposed encryption algorithm.

The rest of the paper is organized into four sections as follows. Section 2 discusses different existing symmetric encryption algorithms for WSNs. We present the network and threat models for our proposed algorithm and give some preliminaries about the Feistel structure in Sect. 3. Section 4 will present our proposal algorithm with its security analysis in Sect. 5. Finally, Sect. 6 shows the performance evaluation of our algorithm, before concluding in Sect. 7.

2 Related works

Cryptography in WSNs will keep secret the information transmitted through networks, but it requires technical ensuring distribution, establishment and management of cryptographic keys used in the encryption/decryption operation. Many papers have proposed symmetric encryption algorithms based on Feistel structure. Here, we discuss some of these algorithms for secure data transmission in WSNs. Specifically, we discuss the concept and implementation and highlight their issues and drawbacks.

One of the symmetric encryption algorithms is the DES [4] which is based on Feistel structure. It splits the data into two parts: left (L) and right (R), and it uses the same transformation function at each round. Note that this function does not have to be reversible. It is here composed of an expansive permutation, the addition of a round key, the passage through eight S-boxes and finally a simple permutation. The DES algorithm does not support any changes in Feistel structure; therefore, it not allows flexibility in it. One of the symmetric block cipher is RC5 [12]. RC5 uses a key size up to 2040 bits and variable sized blocks, 32, 64, 128 bits with a number of rounds up to 255. RC5 is suitable for WSN applications, and can provide a good overall performance when used in nodes with limited processing capabilities and memory. But, the key schedule must be calculated based on 104 additional bytes of RAM. For transmission of secure data in WSNs, Skipjack is another block cipher algorithm [16]. Skipjack uses a 64-bit-sized block with an 80-bit key length, and the selected mode of operation is CBC-CS using a 32 rounds unbalanced Feistel structure. Disadvantage of Skipjack lies in the high energy and an inadequate key length. Madhumita Panda analyzed AES [19] by applying many complex mathematical calculations to the initial message with a 128-bit key using 10 rounds. It supports different combinations of couple (key size, block size). Encryption starts with the formation of a matrix containing the plaintext and a function dedicated to the formation of derived parts from the encryption key. To get a more complex cipher, the authors include for this analysis an implementation of ECC algorithm. But, it is complex to

implement AES on small blocks. Novelan et al. [18] gave the implementation of the encryption algorithm TEA. The encryption process is built on variable rounds Feistel network rounds with a 64-bit-sized block and 128-bit key length. TEA adopts a simplified scrambling function and short data packet, which can greatly reduce the cost of the decryption and encryption. But the algorithm presents a poor key agility (key agility means that the amount of time from generating/importing a new key to starting encrypting is negligible) and increases the burden of the gateway node. TEA has the latest variation XTEA [11]. XTEA is an encryption algorithm based on ECC with a 128-bit key, 64-bit-sized block, and 64 rounds Feistel structure. ECC is used to generate private and public keys for sensors. XTEA greatly improves the lifetime network, but the memory size remains a big challenge for this algorithm. Hong et al. proposed the HIGHT algorithm [8] based on lightweight block. HIGHT provides it periodically round function on entire plaintext for several iterations. This algorithm provides cost-effective encryption and decryption, but does not provide a flexibility in the substitution S-box.

Table 1 presents a summary of the different symmetric encryption algorithms presented below. Most of these algorithms provide a high calculation cost and require large memory due to the transformations used in different rounds and the number of rounds. These parameters affect the security and the lifetime of networks. When proposing such algorithm, an efficient security must be ensured with a low resource consumption. AES and DES use master keys in key establishment. This reduces the storage of keys in node memory. However, resistance to attack is low. Since the master key can be compromised at any time, the different keys established after deployment using this key can be compromised too. The HIGHT, TEA, and XTEA algorithms are costly in ciphertext generation operations because they use secret keys in order to exchange other secret keys. It is extremely important for critical applications in sensor networks to have a security mechanism that ensures authentication and confidentiality. Optimal security in this type of network is particularly difficult due to the limitation of node resources. Keeping in mind these limitations and the various studies of symmetric encryption algorithms, we propose an efficient secure symmetric encryption algorithm for WSNs which reduces the energy consumption and improves network lifetime among all sensor nodes.

3 System models and preliminaries

This section presents the network model, the threat model, and some preliminaries regarding Feistel structure taken for the proposed algorithm.

3.1 Network model

The network is made up of static sensor nodes (100 nodes) randomly deployed over a constant surface of 100×100 m. The network is homogeneous presented in Fig. 1.

Various assumptions about the network model and sensors are as the following guidelines:

Table 1 Summary of some symmetric encryption algorithms for WSNs

Algorithm	Key size	Block cipher	Algorithm structure	Gaps
DES [4]	56 bits	64 bits	16 rounds Feistel structure	Do not provide flexibility in Feistel structure
RC5 [12]	Up to 2040 bits	32, 64, 128 bits	Up to 255 rounds Feistel structure	Subkeys are calculated based on 104 additional bytes of RAM
SKIPJACK [16]	80 bits	64 bits	32 rounds, unbalanced Feistel structure	Inadequate key length, high energy
AES [19]	128 bits	128 bits	10 rounds Feistel structure	Complex to implement on small blocks
TEA [18]	128 bits	64 bits	variable rounds Feistel structure	Poor key agility
XTEA [11]	128 bits	64 bits	64 rounds Feistel structure	The memory size remains a big challenge for this algorithm
HIGHT [8]	128 bits	64 bits	lightweight block algorithm	Not provide a flexibility in the substitution S-box

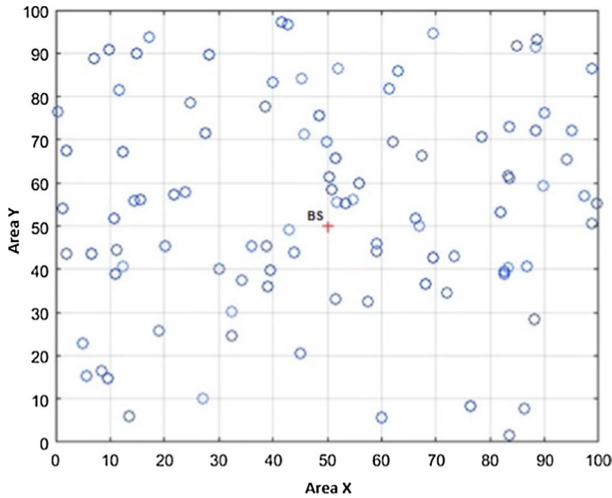


Fig. 1 Network model of proposed algorithm

- The location of each node is unique.
- Each sensor node has a unique identity.
- The base station (BS) is fixed in the center of area [centered in the sensing field (50, 50)].
- All sensor nodes can communicate with each other and the BS.

3.2 Threat model

We assume that an adversary can increase the number of physical attacks on the sensor nodes after the deployment of the network. Once a sensor node is captured, the opponent can read secret information from its memory. Such an attack can compromise not only the adjacent links of those compromised, but also the external links independent of the compromised nodes. The adversary wish to destroy the network with minimum number of sensor nodes. There are several types of attacks that can affect any encryption algorithm. Among these attacks, we cite the following: chosen ciphertext, chosen plaintext, chosen text attacks, ciphertext only, and known plaintext. However, ciphertext only and known plaintext attacks types are the most applicable attacks for symmetric block cipher encryption algorithms. So, we will analyze our algorithm against two attacks:

- Brute force attack: as an example for ciphertext only attacks.
- Differential and linear cryptanalysis: as examples for the known plaintext attacks.

3.3 Feistel cryptosystem structure

Secret key algorithms seek to strive for perfection, which in cryptography is random: it is necessary that encrypted message also appear randomly as possible, to minimize the risk of attack by the analysis of ciphertext. A special case of block ciphers encryption algorithms with iteration is the *Feistel structure* [22]. Figure 2 shows the general process of this structure.

In this encryption, a plaintext block is divided into two halves; the round of transformation is applied to one of these halves, and the result is combined with XOR. This structure provides several advantages, the hardware implementation is also easier, and it is based on simple principles of permutations, substitutions, block data exchanges and function taking as input an intermediate key to each step.

Note $I_n = \{0, 1\}^n$: the data of n bits. So we have $|I_n| = 2^n$. We define an operation on I_n noted \oplus and is called XOR by:

$$(a_1, \dots, a_n) \oplus (b_1, \dots, b_n) = ((a_1 + b_1) \bmod 2, \dots, (a_n + b_n) \bmod 2) \tag{1}$$

Let $f : I_n \rightarrow I_n$ any function, the application is defined $\psi(f)$ of I_{2n} in I_{2n} by:

$$\psi(f)([L, R]) = [P, T] \tag{2}$$

with

$$\begin{cases} P = R \\ T = L \oplus f(R) \end{cases} \tag{3}$$

schematically:

Let f_1, \dots, f_k functions of I_n in I_n , an application $\psi^k : I_n \rightarrow I_n$ defined by:

$$\psi^k(f_1, \dots, f_k) = \psi(f_k) \oplus \psi(f_{k-1}) \oplus \dots \oplus \psi(f_2) \oplus \psi(f_1), \tag{4}$$

is called a *Feistel structure* with k rounds.

Fig. 2 Feistel network structure

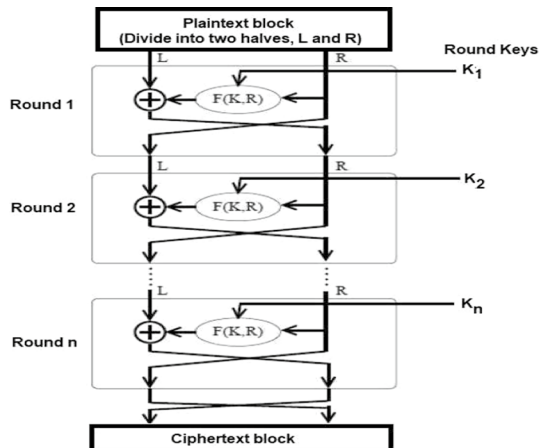


Fig. 3 Example with $\psi^3(f_1, f_2, f_3)$

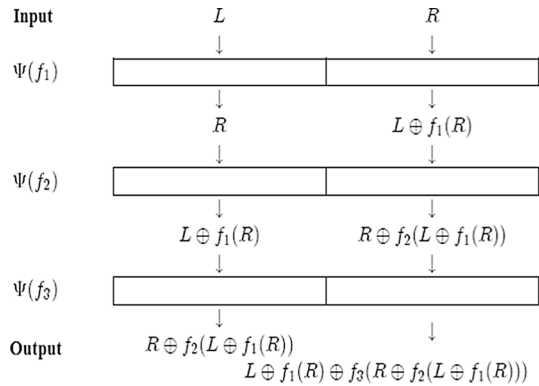
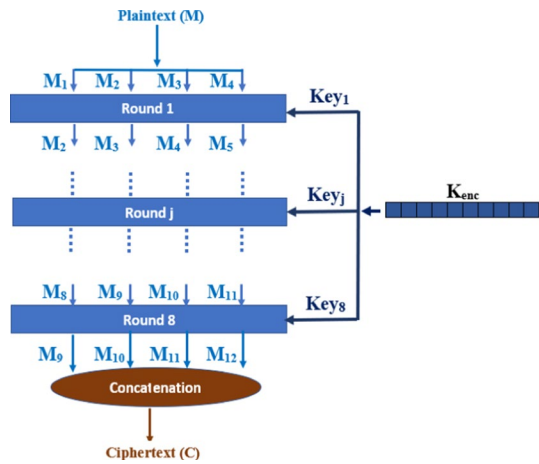


Fig. 4 General ULEA encryption process structure



Example: Examine $\psi^3(f_1, f_2, f_3)$ in Fig. 3.

4 Proposed encryption algorithm

Based on Feistel structure, we propose a novel ultra-lightweight encryption algorithm, named ULEA. It assumes minor encryption rounds, simplified transformations and efficient functions. To reduce the encryption cost of WSNs sensor node, we only execute eight rounds Feistel network with a plaintext of 128-bit and a key input K_{enc} of 256 bits. Figure 4 shows the main steps for our proposed encryption process. We divide the plaintext M into four blocks M_i and K_{enc} into eight subkeys (subkey by each round). After transformations, each round generates four novel blocks in output.

ULEA contains four methods: *key generation, subkeys generation, encryption process and decryption process.*

4.1 Key generation

To encrypt the message, each sensor node generates an encryption key called K_{enc} which shared it with the base station. In this subsection, we present the process of generation of this key which will be used in the ULEA encryption process. Table 2 shows the different parameters used in this key generation process.

Before deployment, the base station pre-distribute to each sensor node of the network a set of keys. The creation of our asymmetric keys is based on the elliptic curves cryptography (ECC) [13]. This cryptography has already proven that it provides a more security of shared keys, which ensures an equivalent level, or even more secure, than other asymmetric systems. The base station begins to create an asymmetric key pair unique to each node in the network. For example, a node N will have a unique key pair (P_N, K_N) . K_N is chosen randomly in an interval $[1, m]$ where m is a parameter of the used elliptic curve. K_N is considered as the private key of N . The public key of N is obtained by the following scalar multiplication:

$$P_N(x_N, y_N) = K_N * G(x, y), \tag{5}$$

where G is the point that lies on the curve.

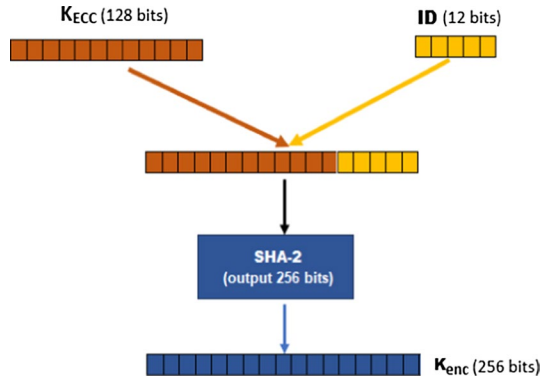
Finally, each sensor node N stores the ECC parameters with its identity ID and the public key of BS, P_{BS} : (P_N, K_N, ID, P_{BS}) . The base station also stores all the public keys of the different sensors in the network.

After deployment of sensor nodes in the network, each node N must create an encryption key, called K_{enc} , used to encrypt message during ULEA encryption process. Our proposed encryption key K_{enc} consists of 256 bits, and contains two components (Fig. 5). The first component K_{ECC} consists of 128 bits produced with the ECC algorithm [13]. The second component ID represents the node identity in 12 bits. After the generation of these components, K_{ECC} and ID are combined and the

Table 2 Key generation parameters

Parameter	Description
N	Sensor node N
BS	Base station
ECC parameters	The different elliptic curve parameters stored on the memory of sensor node
G	Base point that lies on the elliptic curve
ID_N	Identity of sensor node N (12 bits) stored on the memory of sensor node
P_N	Public key of sensor node N (stored on the memory of sensor node)
K_N	Private key of sensor node N (stored on the memory of sensor node)
K_{enc}	The main encryption key (will be generated) with 256 bits
K_{ECC}	Key produced with the ECC algorithm (The first component of K_{enc} with 128 bits)
H	SHA-2 hashing function (output of 256 bits) used to hash the concatenation result of K_{ECC} and ID
t_N	A nonce generated by sensor node N
$MAC_K(M)$	Message authentication code (MAC) of message M using MAC key K
$N \rightarrow BS : M$	Node N sends a message M to BS

Fig. 5 ULEA encryption key



result will be hashed using SHA-2 hashing function (output of 256 bits) [17] to generate our encryption key K_{enc} .

The authentication and key establishment phase is an initial phase that comes right after deployment. Our proposed method uses pre-distributed asymmetric keys before deployment in order to share a secret key between each sensor node and the base station. In our key generation process, we use the Diffie–Hellman mechanism [6] but without interaction between nodes. The idea is that each node in the network can establish a shared secret key K_{ECC} (using ECC parameters) with the base station without interacting with it, that is, without any exchange of messages. The pre-distributed keys in BS and the sensor node N are used to calculate K_{ECC} , which is the first component of K_{enc} , without any exchange. The generation of K_{ECC} is as follows:

BS calculates a temporary key called D_{BS-N} according to Diffie–Hellman:

$$D_{BS-N} = K_{BS} * P_N \tag{6}$$

N calculates a temporary key called D_{N-BS} according to Diffie–Hellman:

$$D_{N-BS} = K_N * P_{BS} \tag{7}$$

We will therefore have

$$\begin{aligned}
 D_{N-BS} &= K_N * P_{BS} \\
 &= K_N * (K_{BS} * G) \\
 &= (K_N * G) * K_{BS} \\
 &= P_N * K_{BS} \\
 &= D_{BS-N}
 \end{aligned} \tag{8}$$

So,

$$K_{ECC} = D_{N-BS} \tag{9}$$

Finally, K_{ECC} and ID are combined and the result will be hashed using SHA-2 hashing function H (output of 256 bits) to generate our encryption key K_{enc} :

$$K_{\text{enc}} = H(K_{\text{ECC}}||ID) \quad (10)$$

Finally, the sensor node N sends the encryption key K_{enc} to BS in a secure manner with an authenticated message (MAC with the private key K_N):

$$N \rightarrow \text{BS} : ID_N || ID_{\text{BS}} || \text{MAC}_{K_N}(K_{\text{enc}}, ID_N || ID_{\text{BS}} || t_N) \quad (11)$$

After this process, K_{enc} will be stored in the memory of the sensor node. With the length of 256 bits, an attacker cannot figure out K_{enc} and this key will keep secret during the encryption process.

4.2 Subkeys generation

With the main encryption key K_{enc} , we generate eight subkeys Key , where the key $_i$ is used in $round_i$. We provide some calculations in this generation process in order to secure the different subkeys against attacks. At initialization, we split the 256-bit main encryption key K_{enc} into 8 blocks K_1, K_2, \dots, K_8 of 32 bits each. The generation of different subkeys Key is based on XOR function between different K and generated Key , with the use of a function $Inv(K)$ which inverse the different bits of K (For example $Inv(10101110101011101010111010100001) = 01010001010100010101000101011110$). The generation details of different subkeys Key are presented in Algorithm 1.

Algorithm 1. Subkeys generation algorithm

- 1- Split the 256-bits main encryption key K_{enc} into 8 blocks of 32-bits: K_1, K_2, \dots, K_8 .
- 2- The 8 subkeys ($Key_1, Key_2, \dots, Key_8$) are computed as follows:

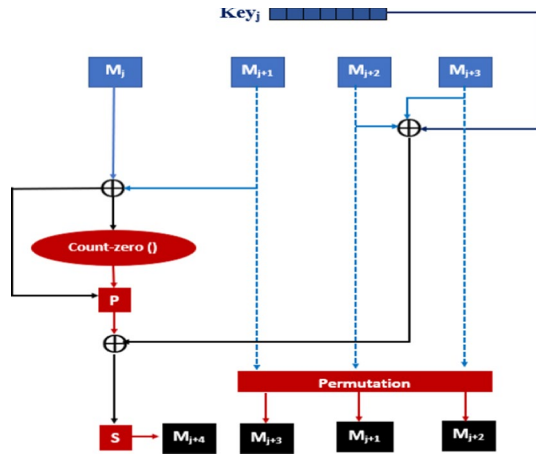
$$\begin{aligned} Key_1 &:= K_1 \oplus [Inv(K_2) \oplus Inv(K_3)] \\ Key_2 &:= Key_1 \oplus [Inv(K_3) \oplus Inv(K_4)] \\ Key_3 &:= Key_2 \oplus [Inv(K_4) \oplus Inv(K_5)] \\ Key_4 &:= Key_3 \oplus Key_2 \oplus Key_1 \\ Key_5 &:= Key_4 \oplus [Inv(K_5) \oplus Inv(K_6)] \\ Key_6 &:= Key_5 \oplus [Inv(K_6) \oplus Inv(K_7)] \\ Key_7 &:= Key_6 \oplus [Inv(K_7) \oplus Inv(K_8)] \\ Key_8 &:= Key_7 \oplus Key_6 \oplus Key_5 \end{aligned}$$

4.3 Encryption process

The contents of each round in the encryption process are shown in Fig. 6. Each round contains four different operations: $Count_zero()$, function P , substitution function S , and permutation. The 128-bit plaintext M is divided into four equal blocks with 32 bits each. Each round generates in output a novel four blocks.

The function $Count_zero()$ calculates the count of 0's for $M_j \oplus M_{j+1}$ and returns the result to function P which used it to modify the value of $M_j \oplus M_{j+1}$ by replacing the 0 by 1. For example, if $M_j \oplus M_{j+1} = 10100011101000111010001110100011$,

Fig. 6 Operations performed by each round j in the encryption process



Count-zero() generates 16, then the function P generates the sequence bits: 1010011101000100010001110100011, which inverse the bits 16 and 17). This result is XORed with the value of concatenation of key_j , M_{j+2} , and M_{j+3} and the final result is transferred to function S , which inverse the different bits (For example: if S receives 10111001101110011011100110111001, it generates 0100011001000110010001101000110).

The encryption algorithm is presented in Algorithm 2.


Algorithm 2. Encryption process

- 1- Divide $M \rightarrow M_1, M_2, M_3, M_4$
- 2- Generate different Key_j from the main key K_{enc}
- 3- for $j=1$ to 8 // number of rounds
 - a) $X := M_{j+2} \oplus M_{j+3} \oplus Key_j$
 - b) $Y := M_j \oplus M_{j+1}$
 - c) $Count := Count-zero(Y)$
 - d) $W := P(Count, Y)$
 - e) $R := W \oplus X$
 - f) $T := S(R)$
 - g) $M_{j+4} := T$
 - h) Permutation: $(M_{j+1}, M_{j+2}, M_{j+3})$
- end for.
- 4- $C := M_9 || M_{10} || M_{11} || M_{12}$ //Ciphertext
- 5- Output C

An example of the function P is presented in Fig. 7.

Finally, the permutation operation for the three blocks M_{j+1} , M_{j+2} and M_{j+3} is used to change the order of the mutated text. The permutation algorithm is presented in Algorithm 3.

Fig. 7 An example of function P operations

M_j	00110101001101010011010100110101
M_{j+1}	01001100010011100100110001001110
$Y = M_j \oplus M_{j+1}$	0111100101111011011100101111011
Function Count-zero () (number of 0's bit in Y)	10
Function P (Inverse the content of bits 10 and 11)	0111100101111011011100101111011 <div style="text-align: center;">  </div> 0111100100011011011110010111011

Algorithm 3. Permutation function for round j

- 1- Let t_r a temporary variable.
- 2- $t_r := M_{j+2}$
- 3- The permutation operations are the following:
 - $M_{j+2} := M_{j+3}$
 - $M_{j+3} := M_{j+1}$
 - $M_{j+1} := t_r$

At the end of all rounds (the end of round 8), the generated ciphertext C is defined as follows:

$$C = M_9 || M_{10} || M_{11} || M_{12} \tag{12}$$

Figure 8 shows an example of a round j in the ULEA encryption process.

4.4 Decryption process

When the base station receives the different ciphertexts, it separates them and the decryption process will start. The base station will use the shared key, K_{enc} , with the sensor node to decrypt the ciphertext and get the plaintext. The general decryption process is presented in Fig. 9. The decryption is the inverse process of encryption. It uses the same functions, with their parameters, presented in the encryption process. There is not essential difference for the encryption/decryption calculations but using the sequence of subkeys Key_j is reversed. The details of the main steps of each round in the decryption process are presented, in Fig. 10. We assume that we know the order of permutation of the blocks realized in the encryption process. The algorithm of this process is presented in Algorithm 4.

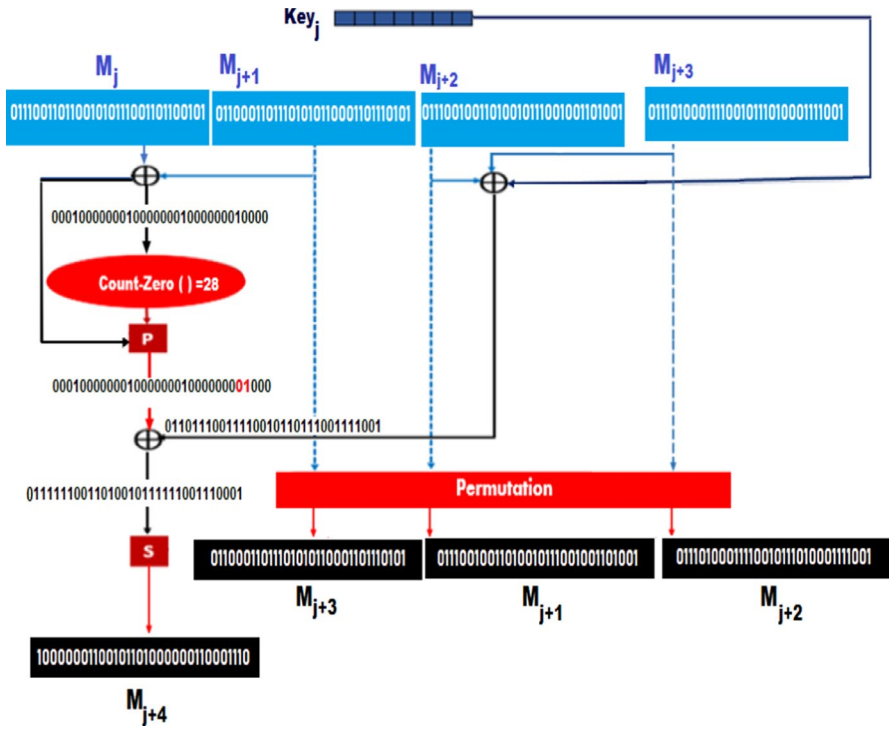
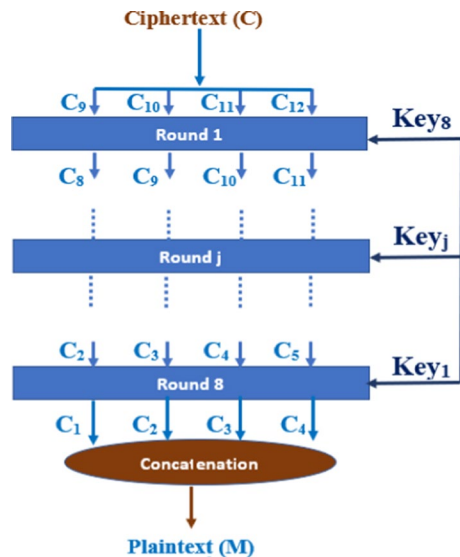


Fig. 8 An example of round j in the encryption process

Fig. 9 Decryption process



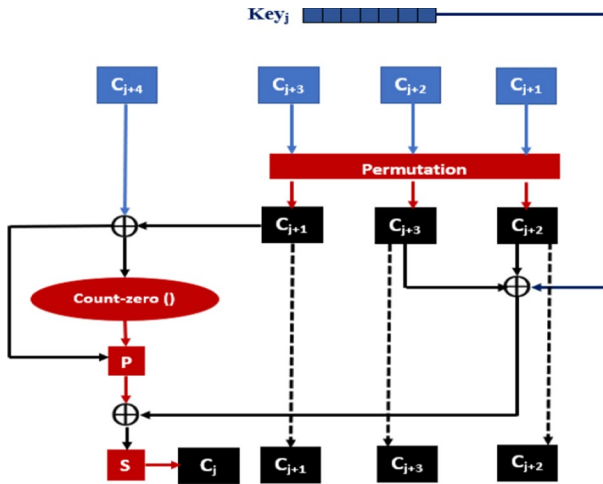


Fig. 10 Operations performed by each round j in the decryption process

Algorithm 4. Decryption process

```

1- Receiving C
2- Generate the 8 sub-keys  $Key$  from the main key  $K_{enc}$  (Algorithm 1)
3- Partition C into 4 segments  $C_9, C_{10}, C_{11}$  and  $C_{12}$ 
4-  $i:=8$ ; //Counter for sub-keys indices
5- for  $j=8$  to 1
    a) Permutation:  $(C_{j+1}, C_{j+2}, C_{j+3})$ 
    b)  $X:=C_{j+2} \oplus C_{j+3} \oplus Key_i$ 
    c)  $Y:=C_{j+4} \oplus C_{j+1}$ 
    d)  $Count := Count-Zero(Y)$ 
    e)  $W:=P(Count, Y)$ 
    f)  $T:=S(W \oplus X)$ 
    g)  $C_j:=T$ 
    h)  $i:=i-1$ 
end for.
6-  $M:= C_1||C_2||C_3||C_4$  //Plaintext
7- Output M
    
```

The permutation operation in the decryption process is presented in Algorithm 5.

Algorithm 5. Permutation function for round j in decryption process

- 1- Let d_r a temporary variable.
 - 2- $d_r := C_{j+1}$
 - 3- The permutation operations are the following:
 - $C_{j+1} := C_{j+3}$
 - $C_{j+3} := C_{j+2}$
 - $C_{j+2} := d_r$
-

Finally, the plaintext M is defined as follows:

$$M = C_1 || C_2 || C_3 || C_4 \quad (13)$$

5 Security analysis

To analyze the security of our proposed algorithm, we describe and discuss some important security analysis results including security requirements, diffusion and confusion, and security against attacks.

5.1 Security requirements

5.1.1 Authentication

Our encryption algorithm provides authenticated messages based on secure key encryption hashed with the SHA-2 (output 256 bits) hashing function H . Indeed, the ECC parameters are combined with the ID to construct an authenticated hash key shared with the base station. According to this process, an attacker cannot figure out K_{enc} which will keep secret during the encryption process and so the base station can verify the authenticity of the source of the transmitted encrypted data.

5.1.2 Confidentiality

In our algorithm, the confidentiality is shown thanks to the use of Feistel rounds encryption. In ULEA, each sensor node encrypts the plaintext based on eight rounds containing efficient operations. So, an attacker cannot recover the ciphertext C and the plaintext M , because he does not have information about the formation of different subkeys. An attacker cannot understand anything when analyzing the ciphertext due to the complex transformations provided by ULEA.

5.1.3 Integrity

Data integrity guarantees that data or information being transferred is never corrupted during data transmission and storage process.

Denote $PE(y)$ the probability of that original data sensor node being compromised; y is referred to the probability of the encryption key being disclosed. Assuming that the attacker knows that the data is transmitted in successive data packets, the probability of that the attacker isolate the target data from the intercepted packets is $q = 1/2$. Given that the maximum node degree is L_{max} , the probability of the node degree being L is $PE(deg = L)$. Denote N the total number of nodes in the network and $N(deg = L)$ is the number of nodes whose degree is L , we have

$$\begin{aligned} PE(y) &= q \times \sum_{L=1}^{L_{max}} PE(deg = L) \times y_L \\ &= \frac{1}{2} \times y \sum_{L=1}^{L_{max}} \frac{N(deg = L)}{N} \times y_L \end{aligned} \quad (14)$$

According to Formula 14, when y is less than 2%, namely

$$\sum_{L=1}^{L_{max}} PE(deg = L) \times y_L < 2\% \quad (15)$$

The probability that original data being compromised is $PE(y) < 0.02\%$ and the data integrity is provided by ULEA.

5.1.4 Availability

Availability ensures that the network is alive and that data is accessible. Moreover, ULEA does not consume more resources in different phases of encryption/decryption processes. In addition, our encryption algorithm assumes minor encryption rounds, simplified transformations and efficient functions to reduce the encryption cost of WSNs sensor node. In addition, with the use of the different processes during our algorithm, we have conserved energy and increased the lifetime of the network. So, ULEA ensures a reasonable level of data availability in the network unlike other algorithms that do not consider the data availability even when a strong adversary exists.

5.2 Diffusion and confusion

Claude Shannon [23] discussed two properties that a good encryption algorithm should check. These are what he calls diffusion on the one hand and confusion on the other. The broadcast property requires that each part of the cipher depends on each part of the plaintext and the key; in other words, small changes in input must have a significant effect on output. The confusion, on the other hand, must serve to make the dependence between plaintext, key and ciphertext more complex. In terms of confusion, there are two basic ways to get it in practice. On the one hand, use the usual non-linear operators in hardware or its bit-by-bit version in software, or even use the modular addition. We can on the other hand use functions with the more complex algebraic expression but in tabulated form, we speak then of S-box. An

S-box is usually chosen to maximize the confusion it causes. Here is where the diffusion layer comes in to mix the outputs of several S-boxes acting on different parts of the encryption. Diffusion, on the other hand, is in linear practice. This makes it possible to use the tools of linear algebra and code theory in order to build good diffusion components, allowing for example to mix between them with the outputs of different S-boxes.

5.2.1 Diffusion

ULEA satisfies the diffusion requirement by the uses of minor encryption rounds, simplified transformations and efficient functions. Each round uses a separate subkey generated from the global encryption key K_{enc} . Moreover, the three functions of 32 bits each generate four blocks in parallel immediate after each round with new values of S-boxes, and make double the bit dependencies. In ULEA, the output blocks lift the dependencies of at least 32 key bits and 32 block bits (the plaintext is of 128 bits divided into 4 blocks), which expedite the rate of diffusion. After one round operation each block bit depends on the previous block bit. Hence, it can be assumed that after successive two ULEA rounds, each intermediate output cipher blocks depends on all the plaintext bits. So, we can confirm that ULEA provides a complete diffusion after two successive rounds in the encryption process.

5.2.2 Confusion

Confusion must serve to make the dependence between plaintext, key and ciphertext more complex, in order to make the statistical work of an attacker more complex. ULEA achieves a higher degree of confusion because of the reasons, as follows:

- In the final of each round, the obtained results are a higher degree of non-linearity because the function *Count-zero()* depends on XOR, the function *P* depends on *Count-zero()* function, and the function *S* depends to function *P*.
- Each round produces an output bit that depends on at least 128 text bits and 32 subkey bits.
- The input-output relation is more complex thanks to the use of efficient transformations and functions.

5.3 Security against attacks

This subsection discusses the proposed algorithm in terms of its security against the two attacks presented in our threat model: brute force attack and Differential and linear cryptanalysis.

5.3.1 Brute force attack

Brute force attack [5] is a method that tests, one by one, all possible combinations of a key. It involves revealing the information hidden in the packages. If the data is not

well protected by a good encryption means, it can be easily recovered by the adversary using a simple analysis. To counter this attack, we must use sufficiently large or periodically renewed keys.

Theorem 1 *The generated ULEA encryption key produces immunity against the brute force attack.*

Proof Brute force attack systematically tests all possible keys until a correct key is found. Compared to AES and DES, in ULEA the key length becomes 256 bits, and an attacker will need to test 2^{256} keys to break our algorithm (a performance detail is illustrated in Table 3). For DES, trying 2^{56} different keys was impossible because it would need an infinite time, but with the use of multiprocessor computer an adversary can obtain all the keys in 20 h [7]. Assuming that the key of DES algorithm can be cracked in 1 s by trying all the 2^{56} keys, our encryption key K_{enc} needs 1.83×10^{63} years to be cracked. Thus, it is too hard for an attacker to obtain the plaintext in the decryption process without knowing the values of different parameters created in network initialization phase and used in our encryption process. So, the generated ULEA encryption key produce immunity against the brute force attack. \square

5.3.2 Differential and linear cryptanalysis

Differential and linear cryptanalysis [15] [1] are among the best known attacks against block ciphers. Since their discoveries, many works have been done to establish a link between these two forms of cryptanalysis or to find better ways to resist such attacks like what was done for the AES encryption. The most consensus on this last point consists in counting the minimum number of active S-boxes, that is to say of S-boxes crossed by the differential or linear characteristics throughout the encryption. After, starting from this to estimate the maximum differential or linear probability over the entire encryption. For an S-box, the output modifications can be created from the knowledge of the input modifications, and each output XOR must be obtained with an equal probability for each input XOR. So, if we have an equal

Table 3 Brute force attack performance by key size

Key size (bits)	Number of alternative keys	Time required at 1 encryption/ μs	Time required at 10^6 encryptions/ μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu\text{s} = 35.8 \text{ min}$	2.15 ms
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu\text{s} = 11.42 \text{ years}$	10.01 h
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu\text{s} = 5.4 \times 10^{24} \text{ years}$	$5.4 \times 10^{18} \text{ years}$
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu\text{s} = 5.9 \times 10^{36} \text{ years}$	$5.9 \times 10^{30} \text{ years}$
256 (ULEA)	$2^{256} = 1.15 \times 10^{77}$	$2^{255} \mu\text{s} = 1.83 \times 10^{63} \text{ years}$	$1.83 \times 10^{57} \text{ years}$

probability input/output XOR distribution for an S-box, this later is not attacked by the differential attack.

- **Differential cryptanalysis**

The differential probability of ULEA is calculated by using the following theorem presented in [2]:

Theorem 2 *If X be the minimum number of active S-boxes and \mathcal{P}_Y is the maximum differential probability of all S-boxes in our proposed algorithm ULEA, so the maximum differential probability \mathcal{P} is bounded by \mathcal{P}_Y^X [2].*

Proof The differential probability for the distribution of 32 based S-box operations is 2^{-2} . Such, the difference on the input bit affects the different bits at the end of each round in the encryption process. Therefore, each round in ULEA generates four active S-boxes. Using the previous theorem, 8 rounds ULEA will have differential characteristic probability of

$$\mathcal{P} \leq 2^{-2 \times 4 \times 8} = 2^{-64} \quad (16)$$

So, the number of active S-boxes is enough for resilience against attacks, before the end of the second round. In addition, after the second round, ULEA is more resistant against this attack. Thus, ULEA is strongly resistant to differential cryptanalysis attack. \square

- **Linear cryptanalysis**

ULEA is designed to provide security by finding different structures to increase the number of active S-boxes in each round. In the encryption process, the are executed 32 S-boxes and the correlation e between the output of a linear expression (which is equal to the number of rounds, 8) and the nonlinear function (which equal to the 32 S-boxes in the encryption process) is

$$e = \frac{8}{32} = \frac{1}{4} = 0.25 \quad (17)$$

Denote L the linear probability of one round in ULEA, which is the difference between the $\frac{1}{2}$ and the probability of a linear expression (equal to $\frac{3}{4}$). L is computed as follows:

$$L = \left| \frac{1}{2} - \frac{3}{4} \right| = 2^{-2} \quad (18)$$

So, for each round in ULEA, the minimum number of active S-Box is 4; therefore, the correlation probability of one round ULEA e' is

$$e' = L \times 2^{-4} = 2^{-2} \times 2^{-4} = 2^{-8} < e \quad (19)$$

Hence, ULEA is resistant to linear cryptanalysis.

Table 4 presents a security performance comparison between ULEA and some algorithms presented in this paper. Our algorithm ULEA increases the key space by changing the number of rounds, with some efficient functions and operations, in order to solve the security problems due to various means of attacks.

6 Experimental results and discussion

In this section, we discuss the results obtained with our experiments. To test the performance of our proposed algorithm, we have done many experiments in a variety of ways. The performance of ULEA is analyzed and compared with AES [19], TEA [18], XTEA [11] and HIGHT [8] algorithms. Particular implementations of the different algorithms have been developed for TelosB motes with the TinyOS system [25]. The implementation is carried out in a simulator dedicated to the WSNs, the TOSSIM simulator [14], and its variant PowerTOSSIM [10] with the multi-precision calculation routines TinyECC [9]. To make the comparison, three crucial parameters have been selected: performance efficiency, storage cost and energy consumption.

6.1 Performance efficiency

The primary goal is to measure the encryption time of ULEA and the other algorithms. In our simulation, only the encryption execution times were measured because decryption time is the same as encryption execution time for the different algorithms. We run these algorithms 30 times in the same length plaintext. Figure 11 shows the comparison of total times expended by the five algorithms. We can conclude that the execution time provided by ULEA is much faster than other algorithms. This is due to the use of simple mathematical XOR, scrambling functions, and a less encryption rounds of ULEA to be completed.

Table 4 Security performance comparison

Algorithm	Block size (bits)	Key size (bits)	Number of rounds
AES	128	128	10
TEA	64	128	Variable rounds
XTEA	64	128	64 rounds
HIGHT	64	128	Variable rounds
ULEA	128	256	8

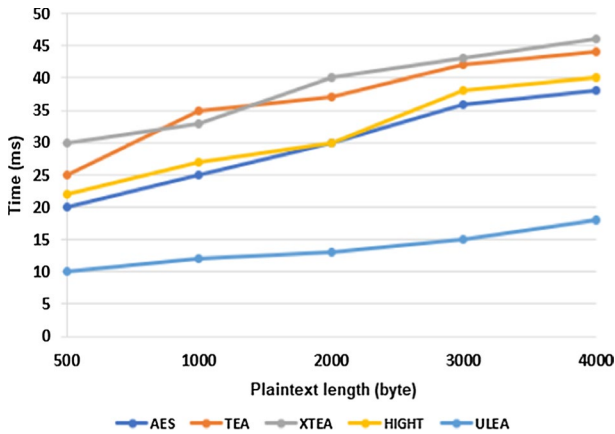


Fig. 11 Performance efficiency comparison

6.2 Storage cost

Figure 12 presents the comparison in terms of occupying memory space result by the five algorithms. During simulation, only the encryption times were measured. TEA algorithm uses the longest S-box and can encrypt data with maximum length of 64 bits, which is present more times than AES and ULEA. XTEA algorithm takes more times than ULEA to execute the encryption process because it contains a more number of rounds. So, ULEA has superiority over the four algorithms.

6.3 Energy consumption

TOSSIM-CC2420 (Simulator provided in the distribution of TinyOS. It models the CC2420 radio) integrates PowerTOSSIM, an extension which models energy. However, TOSSIM cannot provide accurate information on the calculation of the CPU's

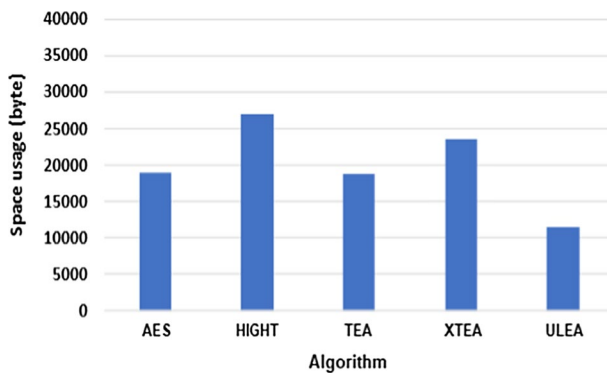


Fig. 12 Storage cost comparison

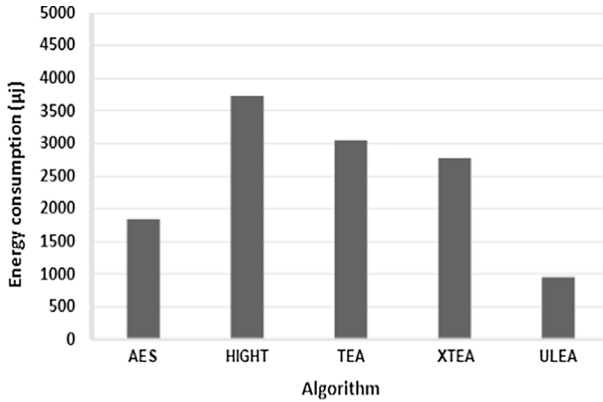


Fig. 13 Total energy consumption depending on encryption algorithm

energy consumption. To do this, we have added to all transmitted/received packets, the corresponding computational load. The energy consumption *Energ* can be calculated by the formula $Energ = V \times F \times T$, where *V* is the tension, *F* is the flow and *T* reflects the execution time. For 2 batteries AA, the tension is 3.0 V. Each simulation takes 500 s. Estimated energy is shown in Fig. 13. It is noted that the total energy consumed increases with increasing number of nodes in the network. ULEA presents a low consumed energy compared to other algorithms, because it presents an efficient operation with less calculation to generate the ciphertext. It is therefore a remarkable energy gain of 65% for ULEA compared to AES.

7 Conclusions and future work

In WSNs, data can be threatened by external events that should not happen during normal network operation. In particular, confidentiality and availability of data are important features that the network should be able to provide. Ensure such characteristics is a difficult task, especially when the nodes have limited hardware capacity.

Based on Feistel structure, we propose in this paper a symmetric encryption algorithm for WSNs. It assumes minor encryption rounds with complex transformations and functions. The security of our algorithm is theoretically analyzed, and our experiments show that the calculation cost is very low with limited memory size and energy consumption compared to other symmetric encryption algorithms. In the following work, we use this algorithm to propose a protocol to secure data aggregation in WSNs, based on homomorphic encryption primitives.

References

1. Biham E, Shamir A (1990) Differential cryptanalysis of des-like cryptosystems. *J Cryptol* 4:3–72. <https://doi.org/10.1007/BF00630563>
2. Budhwar P (2015) Tinyos: an operating system for wireless sensor networks. *J Comput Sci Technol* 6(1):141–145

3. Chen T, Blasco J, Kupwade PH (2019) Cryptography in WSNs, pp 783–820. https://doi.org/10.1007/978-3-319-91146-5_21
4. Coppersmith D (1994) The data encryption standard (DES) and its strength against attacks. *IBM J Res Dev* 38(3):243–250. <https://doi.org/10.1147/rd.383.0243>
5. Dhakne AR, Chatur PN (2017) Detailed survey on attacks in wireless sensor network. In: *International Conference on Data Engineering and Communication Technology*, pp 319–331. https://doi.org/10.1007/978-981-10-1678-3_31
6. Diffie W, Hellman M (1976) New directions in cryptography. *IEEE Trans Inf Theory* 22(6):644–654. <https://doi.org/10.1109/TIT.1976.1055638>
7. Forouzan BA (2008) *Cryptography and network security*. Mc Graw Hill, New York
8. Hong D, Sung J et al (2006) Hight: a new block cipher suitable for low-resource device. In: *Cryptographic Hardware and Embedded Systems—CHES*, pp 46–59. https://doi.org/10.1007/11894063_4
9. Karagiannis A, Vouyioukas D (2013) Performance enhancement of TinyECC based on multiplication optimizations. *Secur Commun Netw* 6(2):151–160. <https://doi.org/10.1002/sec.422>
10. Karagiannis A, Vouyioukas D (2015) A framework for the estimation and validation of energy consumption in wireless sensor networks. *J Sens*. <https://doi.org/10.1155/2015/124987>
11. Kotel S, Zeghid M, Machhout M, Tourki R (2017) Lightweight encryption algorithm based on modified XTEA for low-resource embedded devices. In: *21st International Database Engineering and Applications Symposium*, pp 192–199. <https://doi.org/10.1145/3105831.3105853>
12. Kukkurainen J, Soini M, Sydanheimo L (2010) Rc5-based security in wireless sensor networks: utilization and performance. *WSEAS Trans Comput* 9(10):1191–1200
13. Lauter K (2004) The advantages of elliptic curve cryptography for wireless security. *IEEE Wirel Commun* 11(1):62–67. <https://doi.org/10.1109/MWC.2004.1269719>
14. Li J, Serpen G (2012) Simulating heterogeneous and larger-scale wireless sensor networks with TOS-SIM TinyOS emulator. *Proc Comput Sci* 12(3):374–379. <https://doi.org/10.1016/j.procs.2012.09.089>
15. Matsui E (1993) Linear cryptanalysis method for des cipher. In: *Advances in Cryptology-EUROCRYPT*, pp 386–397. https://doi.org/10.1007/3-540-48285-7_33
16. Milad A, Muda H, Muhamad N, Algaet M (2012) Comparative study of performance in cryptography algorithms (Blowfish and Skipjack). *J Comput Sci* 8(01):1191–1197. <https://doi.org/10.3844/jcssp.2012.1191.1197>
17. National Institute of standards and Technology (2002) FIPS PUB 180-2: Secure Hash Standard
18. Novelan M, Husein A, Harahap M, Aisyah S (2018) Sms security system on mobile devices using tiny encryption algorithm. *J Phys Conf Ser* 1007(4):12–37. <https://doi.org/10.1088/1742-6596/1007/1/012037>
19. Panda M (2015) Data security in wireless sensor networks via AES algorithm. In: *IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, pp 1–5. <https://doi.org/10.1109/ISCO.2015.7282377>
20. Patel R, Kumar S (2018) Wireless sensor networks' challenges and future prospects. In: *International Conference on System Modeling, Advancement in Research Trends (SMART)*, pp 60–65. <https://doi.org/10.1109/SYSMART.2018.8746937>
21. Patel ST, Mistry NH (2015) A survey: lightweight cryptography in WSN. In: *International Conference on Communication Networks (ICCN)*, pp 11–15. <https://doi.org/10.1109/ICCN.2015.3>
22. Rebeiro C, Nguyen PH, Mukhopadhyay D, Poschmann A (2013) Formalizing the effect of Feistel cipher structures on differential cache attacks. *IEEE Trans Inf Forensics Secur* 8(8):1274–1279. <https://doi.org/10.1109/TIFS.2013.2267733>
23. Shannon C (1949) Communication theory of secrecy systems. *Bell Syst Tech J* 8(9):656–715. <https://doi.org/10.1002/j.1538-7305.1949.tb00928.x>
24. SunilKumar K, Shivashankar (2017) A review on security and privacy issues in wireless sensor networks. In: *IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT)*, pp 1979–1984. <https://doi.org/10.1109/RTEICT.2017.8256945>
25. Tripathy S, Sukumar N (2009) LCASE: Lightweight cellular automata-based symmetric-key encryption. *Int J Netw Secur* 8(2):243–252