An efficient cryptographic technique using modified Diffie–Hellman in wireless sensor networks

Shahwar Ali¹, A Humaria¹, M Sher Ramzan², Imran Khan¹, Syed M Saqlain¹, Anwar Ghani¹, J Zakia¹ and Bander A Alzahrani²

Abstract
In wireless sensor networks, the sensors transfer data through radio signals to a remote base station. Sensor nodes are used to sense environmental conditions such as temperature, strain, humidity, sound, vibration, and position. Data security is a major issue in wireless sensor networks since data travel over the naturally exposed wireless channel where malicious attackers may get access to critical information. The sensors in wireless sensor networks are resource-constrained devices whereas the existing data security approaches have complex security mechanisms with high computational and response times affecting the network lifetime. Furthermore, existing systems, such as secure efficient encryption algorithm, use the Diffie–Hellman approach for key generation and exchange; however, Diffie–Hellman is highly vulnerable to the man-in-the-middle attack. This article introduces a data security approach with less computational and response times based on a modified version of Diffie–Hellman. The Diffie–Hellman has been modified to secure it against attacks by generating a hash of each value that is transmitted over the network. The proposed approach has been analyzed for security against various attacks. Furthermore, it has also been analyzed in terms of encryption/decryption time, computation time, and key generation time for different sizes of data. The comparative analysis with the existing approaches shows that the proposed approach performs better in most of the cases.

Keywords
Wireless sensor network, cryptography

Date received: 10 January 2020; accepted: 17 April 2020

Handling Editor: Mohsin Iftikhar

Background
The security of data is a foremost topic in every arena since the data are sent over an unreliable network. Numerous approaches have been proposed to protect data; however, cryptography is one of the most dependable approaches used to secure data. Cryptographic techniques help in altering the original data into incomprehensible data. Two major approaches to cryptography are symmetric key and asymmetric keys. The symmetric key uses only one key for both encryption and decryption of the data, which consumes less energy and memory. The asymmetric key uses two keys, where the private key is used for decryption and the public key is used for encryption. The asymmetric cryptographic approach is much secure than symmetric because it uses two keys. The generation of two keys will cause high energy, time, and memory consumption. Cryptographic algorithms can be in either block or

¹International Islamic University Islamabad, Islamabad, Pakistan
²Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

Corresponding author:
Imran Khan, International Islamic University Islamabad, Islamabad 44000, Pakistan.
Email: imran.khan@iiu.edu.pk

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
stream ciphers category. In wireless sensor networks (WSNs), data travel over sensor nodes, which can be unsafe from various attacks. Various attacks can occur on data in the WSN environment as shown in Figure 1. To protect data, numerous approaches have been published.

Introduction

The WSN is a versatile network as shown in Figure 2, there exist multiple types of sensor nodes: sink node, base station (BS), sensor nodes, and cluster heads depending upon which routing methodology is used. The data are sent from the sensor node to the cluster node which is then forwarded to BS for the advanced communication process. Sensor nodes are used to sense and transmit data in different situations. In many real-time packages, the sensor nodes are performing one of a kind duty like neighbor node discovery, clever sensing, data confidentiality goal monitoring, tracking, node localization, synchronization, and efficient routing among nodes and BSs.1

Secure data transmission is a major issue in WSN since over the network there exist many attackers that can attack or forge the data. This research of WSN data security focuses on research published during the years 2016 to 2019. The number of articles studied from the years 2016 to 2019 is mentioned in Table 1. Furthermore, section “System architecture” gives a detailed description of these articles.

Sensor nodes have limited computational capacity, memory, and power.2 There is a significant amount of research published to handle data security in WSNs. Elliptic curve cryptography (ECC) is used by a variety of research methods proposed.3,4 It is also used in combination with Advance Encryption Standard (AES) by Ullah et al.,5 where the key is generated by ECC. However, AES is a block cipher algorithm, which is high in computational time depending on the size of keys and plaintext. ECC is also combined with deoxyribonucleic acid (DNA),6 to assure secure data communication. The chaotic map is a highly secure statistical method and Rivest–Shamir–Adleman (RSA) is also used for data security in WSNs.7,8 Since the WSN is a complicated environment, few proposed research approaches like AES- hybrid elliptic curve cryptography (HECC)5 and elliptic curve cryptography genetic algorithm (EGASON)3 considered most appropriate.

This article introduced a new cryptographic approach for data security in WSNs. The low-energy adaptive clustering hierarchy (LEACH) protocol is used for data routing. The proposed research approach uses Diffie–Hellman for key generation. Since Diffie–Hellman is an asymmetric approach, it gives high security. This encryption approach has less response and computational time.

The rest of the document is divided as follows: section “Literature review” discusses the Literature Review, where each research paper is discussing in detail. Section “Previous research analysis” discusses the previous research analysis. Section “Motivation” tells about the contribution to this research. Section “Proposed scheme” discusses the proposed approach in detail. Section “Experimental results and discussions” shows the experimental results. Finally, section “SMDH versus Diffie–Hellman” shows the references.

System architecture

The WSN contains various nodes known as sensor nodes which communicate with each other. The communication between two nodes involves a sink node and a BS that help route the data from one sensor node to another. The basic idea of WSNs is route data between nodes wirelessly. Figure 3 shows an overview of WSNs where various nodes such as sink node, sensor node, and BS are displayed.
Literature review

Data security is a major issue in WSNs. The foremost vital challenge to secure information using an algorithm which has less computational time, less response time, low power consumption, and limited bandwidth. A variety of algorithms is planned to secure knowledge to conquered problems such as overcoming lack of power, coverage issues, and creating restricted use of information measure. This section is generalized into multiple sections as follows: an elliptic curve, AES, RSA, chaotic maps, efficient block ciphers, and various other techniques. Figure 4 gives a summary of multiple analysis approaches studied for data security in WSNs.

The analysis in Elhoseny et al.\(^3\) introduced a new technique named as EGASON. This approach uses ECC for a key generation\(^4\) and then the genetic algorithm approach is used for the encryption and decoding of information. While using ECC, this approach may suffer from brute-force attack and high computation cost in comparison with symmetric key functions.\(^9,10\) The use of pseudo-random number generator (PRNG) can be broken with the help of the method in Somsuk and Sanemueang.\(^11\)

Secure encryption is presented in Santhosh and Shalini.\(^12\) In terms of security, this algorithm only uses exclusive or (XOR) operation for the encryption, making it uncertain against the known-plaintext and chosen-plaintext attacks.\(^13\)

Additional research proposed elliptic curve cryptography-key management (E-KM)\(^14\) where a key organizing procedure is presented. This research methodology is insecure against a collision search attack.\(^15\) The latest studies verify that using ECC makes it at risk to invalid curve attacks occur.\(^16\)

The analysis in Viswanathan and Kannan\(^17\) uses elliptic curve key cryptography with beta and gamma functions for secure routing in WSNs. There are many approaches that use ECC for WSNs and which is why there is a need for a new secure encryption algorithm for WSNs.

Research in Ghani et al.\(^18\) used the symmetric cryptography approach for sensor authentication in WSNs and the generation of shared key for encryption and decryption as a new key generation approach. The use of a random number generator in this research can be fragmented by the technique given in Reeds.\(^19\)

A study in Ullah et al.\(^5\) introduced HECC as a new key generation approach. This key supports encryption/decryption and node authentication. The use of a random number generator in this research can be fragmented by the technique given in Reeds.\(^19\) In this method, AES is used to provide data security, making this method susceptible to biclique attacks.\(^20\)

Research in Tiwari and Kim\(^6\) uses ECC and DNA for data encryption and decryption. DNA is castoff to assign genes, and these genes are used for data encryption. This study is protected against timing and simple power analysis (SPA) attacks. However, this procedure is susceptible to man-in-the-middle (MIM) attack.\(^21\)

The research in Suresh and Hegadi\(^22,23\) focused on the IoT environment. Dual cryptography architecture-secure network communication (D-SN) is a data security method that uses DNA genes sequence for encryption with RSA and Data Encryption Standard (DES). The use of a minor public and private key size will be led to a security issue.\(^24\)

The research in De Weger,\(^25\) uses RSA which is divided into four phases: key distribution and generation phase, encryption phase, enroute filtering phase, and routing phase. RSA in this algorithm makes it prone to Wiener and Boneh–Durfee attacks.\(^26\)
Modified Rivest–Shamir–Adleman (MRSA) is an additional encryption approach, which modifies the existing RSA. This research method is insecure against chosen-ciphertext attack. Moreover, energy consumption also increases because of three prime numbers.

Novel research in Babu and Balasubadra uses elliptic-curve Diffie–Hellman–Key Extraction (ECDHKE) procedure to deliver end-to-end confidentiality, guaranteeing authentication, and integration. However, the use of RSA makes this approach susceptible to chosen-ciphertext attack. ECDH makes it vulnerable to MIM attack.

Elliptic Curve Cryptography—Advance Encryption Standard (E-AES) is an additional study approach that uses ECC to generate keys for encryption and decryption. This procedure is very secure, however, has high complexity and large communication overhead. Another research in Li provides an enhanced version of AES. However, AES is prone to biclique attacks. The key used in this method is uncertain from related-key attack.

Advance Encryption Standard–Quadrature Phase Shift Keying (AES-QPSK) without and with low-density parity check (LDPC) is one more study approach that uses AES. There is a possibility that this procedure insecure against biclique attacks. This study method is also susceptible to related related-key attack. Another research uses AES for data encryption, with a mixture of mutation technique. This procedure is not safe against the differential attacks.

Additional research in Vangala and Parwekar uses mutation with AES for the generation of factors and initial seed. This approach is highly apprehensive about biclique attacks. PRNG makes this procedure susceptible to direct cryptanalytic attack, input-based attack, backtracking attack, and several more. PRNG can be broken down using the method in Reeds.

Hybrid chaotic cryptosystem tent piecewise linear chaotic map (HTPW) is an innovative research method for key managing which uses skew and the maps. This method is insecure to chosen- and known-plaintext attacks because only XOR procedure is used in terms of security, enhanced secure data using chaotic-based encryption (EDCB) is an additional method for securing data in WSNs. This method uses chaotic maps for the generation of keys. Chaotic maps can surely produce unique numbers, but the use of chaotic maps makes this technique apprehensive against ciphertext only, known plaintext, chosen plaintext and chosen ciphertext.

The study in Nidarsh and Devi practices LEACH protocol for data routing and then uses chaotic maps for the data encryption and decryption. Chaotic maps are breakable from numerous methods in Sobhy and Shehata. Another research uses logistic and Kent chaotic maps. This methodology has few drawbacks containing weak keys, fixed chaotic sub-matrices, and plain-image insensitivity.

A lightweight block cipher (QTL) is an ultra-lightweight block cipher approach presented in Li. However, this approach is not secure from differential and linear attack. Linear cryptanalysis (LC) is another ultra-lightweight block cipher approach, presented in Patil et al. This algorithm is insecure against known-plaintext attack and chosen-plaintext attack.

A new lightweight algorithm lightweight pseudo-random number generator (LGA) is introduced in Maity et al. With the use of permutation, it is possible to launch a known/chosen-plaintext attack on this technique. Another research in Solomon and Elias introduces ciphertext-policy attribute-based (C-AB) encryption lightweight encryption and authentication code generation scheme. Data encryption with this algorithm, secure data against eavesdrop attack. Secure hash algorithm (SHA)-3 may get effected by differential attack. Decision support system (DSS) is not secure from to timing attack.

The research in Praveena presents Ultra-Encryption Standard Version 4 (UES-4) scheme. Numerous present cryptographic techniques are united to produce UES-4. Bit-wise reorganizing is performed on the plaintext, then bitwise columnar transportation is performed for unreadable text. In the meantime, multiple encryptions are performed; it becomes very tough to predict the plaintext, making UES-4 safe against brute-force attack. This method is not secure against ciphertext-only attack (COA).

Low complexity security algorithm (LSA) is a different data security research method. The progress of the nodes is the first step. The key is XORed with the plaintext. Since nodes are only XORed, they take less computational time. XOR has a weak defense against brute-force attack.

This research presents Modified Rotation XOR (MR-XOR), which is an altered version of simple XOR. XOR operation is vulnerable to brute-force attack. Research in Praveena and Smy proposed Modern Encryption Standard-Version 4 (MES-IV) using Trisha, Tomodep, Jayshree, Shayan and Asoke (TTJSA) and Dripto, Jayshree, Soumitra, Suvadeep, Asoke (DJSA) for the key generation. This procedure has strong protection against known-plaintext, brute-force, and differential attacks. This research method is prone to side-channel attacks since the ciphertext can be attained.

This study presents digital signature based on key management (DK) for data security. Asymmetric method doubles the computation time.

Research in Gracy and Venkatesan uses honey encryption (HoneyE) procedure to create misperception. A key
that this algorithm provides a better result than the existing algorithm.

**Previous research analysis**

Based on research done in past years, it is considered that this research can be analyzed from the perspective of security. While most of the given approaches prevent from multiple attacks. The security analysis of the studied literature is given in Table 2. The table compares the literature in terms of data security and routing protocol. Manifold deliberate literature does not give both routing and data security approach.

Table 3 briefly discusses multiple attacks that can be prevented using different approaches. Even after protecting the data against multiple attacks, these mentioned approaches suffer from multiple disadvantages. The research in Ullah et al.\(^5\) uses AES for encryption of the data, and keys are generated through HECC. The given approach is secure from forward and backward–secrecy; however, using the Random number generator, this approach can be broken. The use of AES makes this approach vulnerable to biclique attacks. Another approach in Manger\(^27\) modifies the RSA where three prime numbers are used instead of two, which makes brute-force attack difficult to occur. The use of RSA, however, makes this approach prone to chosen-ciphertext attack.

**Motivation**

Data security in WSNs is a major issue, where the security is low which led to related-key, plaintext, or MIM attacks. EGASON\(^1\) could be affected by a plaintext attack and MIM attack. Another major issue of data security in WSNs is that the existing data encryption approaches have either very high response time or high computational time. AES-HECC\(^54\) has high computational time and EGASON\(^1\) led to high response time. A new secure algorithm that will consume less computational time and less response time is introduced. Based on the above literature, a new approach for data encryption is proposed.

**Proposed scheme**

In this section, system architecture and proposed schemes are discussed. First, secure efficient encryption algorithm (SEEA) is discussed. SEEA uses secure modified Diffie–Hellman (SMDH) approach for key generation, which is then used in SEEA. The data are encrypted using SEEA and SMDH, then that data are routed with LEACH. All these approaches are discussed in detail in the following. A notation table is given in Table 4.
| Serial no. | Technique name | Attack prevention | Criticism |
|-----------|----------------|-------------------|-----------|
| 1         | MES-2<sup>59</sup> | Prevent from known-plaintext, brute-force, and differential attacks | Side-channel attack can occur<sup>60</sup> |
| 2         | DK<sup>61</sup> | – | Vulnerable to related-key attacks<sup>73</sup> |
| 3         | RC5-CBC<sup>74</sup> | – | Differential attack and linear attack occur because of the usage of RC5,<sup>75</sup> also used PRNG, which makes it vulnerable to direct cryptanalytic attack, input-based attack, backtracking attack and many more<sup>38</sup> |
| 4         | HTPW<sup>39</sup> | – | Chosen-plaintext attack can occur on this because of XOR operation used. Also suffer from known-plaintext attack<sup>3</sup> |
| 5         | EGASON<sup>3</sup> | Passive attack, key-space analysis attack, compromised CH attack, Sinkhole attack, HELLO flood attack, and DOS attacks cannot occur | Vulnerable to brute-force attack. The intruder can use the short period of PRNG<sup>11</sup> to generate private keys, as this technique allows to generate random number (key)<sup>19</sup> |
| 6         | Q-T-L<sup>54</sup> | Prevent from differential, linear, Algebraic and related-key attacks | Even after claiming, the proposed algorithm suffers from differential and linear attacks<sup>45</sup> |
| 7         | UES-4<sup>53</sup> | Protect against brute-force attack | Ciphertext-only attack can occur<sup>54</sup> |
| 8         | LC<sup>46</sup> | Linear, differential, zero correlation and bilique attacks cannot occur | Suffer from known-plaintext and chosen-plaintext attacks |
| 9         | LSA<sup>76</sup> | – | Brute force can occur<sup>56</sup> and can also suffer from known-plaintext attack<sup>28</sup> |
| 10        | EDCB<sup>80</sup> | – | Ciphertext-only, known-plaintext, chosen-plaintext, and chosen-ciphertext attacks can occur<sup>41</sup> |
| 11        | M-AES<sup>32</sup> | – | Because of the usage of AES, therefore biclique<sup>20</sup> and related-key attacks can occur<sup>33</sup> |
| 12        | AES-QPSK without and with LDPC<sup>34</sup> EEM<sup>35</sup> | – | Can suffer from biclique,<sup>20</sup> related-key attacks<sup>33</sup> |
| 13        | Known-plaintext, chosen-plaintext, chosen-ciphertext, brute-force and ciphertext-only attacks cannot occur | High risk of differential attacks<sup>26</sup> |
| 14        | LGA<sup>48</sup> | – | Usage of randomly choosing method, known- and chosen-plaintext attacks can occur<sup>9</sup> |
| 15        | LED<sup>77</sup> | Replay, eavesdropping and known-key attacks can occur | Because of SHA-1, it can suffer from collision search attacks<sup>78</sup> |
| 16        | SCM<sup>12</sup> | – | Using XOR, makes known-plaintext and chosen-plaintext attack occur<sup>13</sup> |
| 17        | D-SN<sup>22</sup> | Brute-force attack cannot occur | The following paper shows the procedure of how the RSA is attackable.<sup>79</sup> Small public and private key are can also occur as given in Biham<sup>90</sup> |
| 18        | E-KM<sup>14</sup> | – | Because of ECC collision attack occur.<sup>15</sup> And invalid curve attack can also occurs<sup>16</sup> |
| 19        | ECB PU<sup>81</sup> | – | Vulnerable to chosen-plaintext attack<sup>82</sup> |
| 20        | M.XOR<sup>77</sup> | – | This proposed algorithm uses PRNG, to randomly generate number uniquely, which make this algorithm suffer from direct cryptanalysis attack, input-based attack, meet-in-the-middle attack, and many more, as proved in Kelsey et al.<sup>38</sup> |
| 21        | C-AB<sup>50</sup> | Eavesdrop does not occur | SHA-3 can get effected from differential attack<sup>51</sup> |
| 22        | E-AES<sup>83</sup> | – | The problem occurred in the planned procedure is security, high complexity and has high communication overhead, as mentioned in Ullah et al.<sup>5</sup> |

(continued)
The proposed approach is divided into three phases. Each phase is designed to have less computational and response time. The three phases designed are given below.

**Phase 1.** SMDH is part of SEEA.
  a. Simple efficient hashing (SEH).

**Phase 2.** SEEA.

**Phase 3.** LEACH.

LEACH for WSNs is introduced to provide secure data communication between two parties. LEACH delivers secure communication between sensor nodes and protects from various attacks.

**SEEA**

SEEA is a new efficient encryption algorithm for WSNs. This method is designed in terms of less computational and less response time. Since the proposed approach does not require additional complex steps, this gives advantages in reducing the computational time. The proposed approach is divided into two phases: in the first phase, the key is generated and in the second phase, the proposed encryption algorithm is applied.

**Phase 1:** key generation

In the key generation phase, two parties who want to build communication decide a common $p$ and $g$ value,
which are prime numbers. The values are then used in a formula: \( G \mod P \). After \( P \) and \( G \) are known by both the parties, they both use the formula to generate a new value, which is passed to both the parties with the help them generate their private key with their secret values. The working of the original Diffie–Hellman Algorithm is discussed in Diffie and Hellman.85 Since Diffie–Hellman is highly vulnerable to MIM attack, an improved version of Diffie–Hellman is introduced known as SMDH.

**SMDH**

MIM attack is a very common attack over the network, where data can be easily changed. SMDH helps prevents MIM attack in Diffie–Hellman. The generated key of any encryption needs to be secure unless the security of any algorithm will fall apart.

Hashing is used for the longest time to provide secure communication between communicating parties. The basic concept of hashing is to make the original data convert into a format that is meaningless and difficult to break. Only the communicating parties can break the hashing, which allows hashing to work as an authentication method. The hash value can only be calculated if the right method and values are used, and only the communicating parties are able to perform this task.

SMDH approach modifies Diffie–Hellman using a hashing function to overcome the MIM attack. The receiver is the only person who can generate correct hashing value due to the knowledge of knowing

| Serial no | Technique name | Data security | Routing | Comments |
|-----------|----------------|---------------|---------|----------|
| 1         | MES-259        | ✓             | ✓       | Has secure key |
| 2         | DK51           | ✓             | ✓       |           |
| 3         | RCS-CBC74      | ×             | ✓       | Decrease energy |
| 4         | HTPW79         | ✓             | ✓       | Proves to be better than the SPIN algorithm |
| 5         | EGASON3        | ✓             | ✓       | Better than RCS5 |
| 6         | QTL44          | ✓             | ✓       | Less key and plaintext size |
| 7         | UES-433        | ✓             | ✓       | Faster than Picolo, PRESENT, MIBS |
| 8         | LC66           | ✓             | ✓       |           |
| 9         | LSA76          | ✓             | ✓       | Less memory consumption |
| 10        | EDCB10         | ✓             | ✓       | Less time consuming |
| 11        | M-AES32        | ✓             | ✓       | Better performance than skipjack |
| 12        | AES-QPSK without and with | ✓ | ✓ | Improved AES version for WSNs |
| 13        | EEM35          | ✓             | ×       | Less time, yet high security |
| 14        | LGABG8         | ✓             | ×       | Better than existing AES |
| 15        | LED77          | ✓             | ✓       | Better key selection |
| 16        | SCM12          | ✓             | ✓       | Better than AES |
| 17        | D-SN22         | ✓             | ✓       | Less throughput and delivery rate |
| 18        | E-KM14         | ✓             | ×       | Better in response time, security and scalability than existing methods |
| 19        | ECB_PL81       | ✓             | ✓       | Better ECC results |
| 20        | M.XOR27        | ✓             | ×       | Time consumption is high |
| 21        | C-AB10         | ✓             | ×       | Better throughput and total overhead rate |
| 22        | E-AES53        | ✓             | ✓       | Lightweight encryption |
| 23        | AES-Mutation37 | ✓             | ✓       | Very secure and efficient |
| 24        | Modified RSA25 | ✓             | ✓       | Less energy consumption |
| 25        | MRSA27         | ✓             | ✓       | Performance is better than BECAN |
| 26        | Chaos92        | ✓             | ✓       | High security than RSA |
| 27        | HoneyE65       | ✓             | ✓       | Healthier than low-energy adaptive clustering hierarchy (LEACH) in terms of lifetime, throughput, packet delivery, and total energy used |
| 28        | ECDH-KE30      | ✓             | ×       | Better than DPBSV and AES |
| 29        | AES-HIECC3     | ✓             | ✓       | Less energy consumption, and communication cost |
| 30        | XOR-Chaotic maps7 | ✓ | ✓ | Less computational cost |
| 31        | Chaotic data integrity and homomorphic data encryption79 | ✓ | ✓ | Better performance |
| 32        | DNA-E8         | ✓             | ×       | Performance is better than MD5 |

WSN: wireless sensor network; ECC: elliptic curve cryptography; DPBSV: dynamic prime number based on security verification.
correction function and parameters. To calculate the hashing value, an efficient hashing function is introduced.

All values in Diffie–Hellman, such as \( P_1, P_2, P_3 \), and \( G \), are insecure. Before passing these values over the network, a hash value is calculated. Any value (e.g. \( P_1 \)) is first converted into binary form. After this conversion number of 1’s is calculated and stored that number into a temporary variable such as Temp. The mod is then calculated of \( P_1 \) and Temp. The outcome from mod value is attached to the original \( P_1 \) binary value. To separate the hash value from the original \( P_1 \) value, the number of zero’s is calculated. The number of zeros will help to add that specific amount of special character in between the original \( P_1 \) value and hash value. This minor hash value calculation will help to protect those flaws in Diffie–Hellman, and at the same time keeping the computation time low. The algorithm of SMDH is given in Table 5. In Figure 4, an overview of the modified Diffie–Hellman approach is given. The proposed approach will help to prevent MIM attack. The following steps discuss the working of the SMDH approach.

1: Person A and Person B decide two common parameters, known as \( P \) and \( G \).

2: \( P \) and \( G \) are send using the hash function, as follows:
   a. Convert \( P \) and \( G \) values into binary.
   b. Count the number of 1’s in the converted binary values.
   c. Store the count value into a variable.
   d. Take the mod of \( P \) and Count value.
   e. Then count the number of zero's.
   f. Add characters of count value gained in the step above.
   g. Then write the original \( P \) binary, add special character, and finally add the value of step d at the send.
   h. The same process follows for \( G \).

3: Then send this value to Person B.

### Table 4. Notation table.

| Symbol | Description |
|--------|-------------|
| \( e \) | Plaintext (PT) |
| \( e' \) | The ciphertext (CT) |
| temp \(_1\) | A. Temporary 1 CT |
| temp \(_2\) | i) Temporary 2 CT |
| temp \(_3\) | a) Temporary 3 CT |
| b) \(<\) | c) Left shift 1 |
| d) \(\geq\) | e) Swapping data |
| f) Set1 | g) Subset 1 |
| h) Set2 | i) Subset 2 |
| temp \(_1\) \(_a\) | (i) Temporary 1 PT |
| temp \(_2\) \(_a\) | (a) Temporary 1 PT |
| temp \(_3\) \(_a\) | (i) Temporary 1 PT |
| (ii) CH | (iii) Cluster head |
| (iv) N1 | (v) Node 1 |
| (vi) N2 | (vii) Node 2 |
| (viii) SN | (ix) Sink node |
| (x) PU | (xi) Public key |
| (xii) PR | (xiii) Private key |
| (xiv) P | (xv) Parameter (value) |
| (xvi) Q | (xvii) Parameter (value) |
| (xviii) K | (xix) Key |

### Table 5. Secure modified Diffie–Hellman.

Key generation (using secure modified Diffie–Hellman)

| Step | Description |
|------|-------------|
| 1.   | Two communicating parties select common \( P \) and \( G \) |
| 2.   | \( h(P) \) and \( h(G) \) are calculated |
| 2.1  | \( P \) and \( G \) value convert to binary |
| 2.2  | \( Temp = Calculate number of 1's in P \) and \( G \) (e.g. \( Temp = 3 \)) |
| 2.3  | \( P = (P \mod Temp) \) and \( G = (G \mod Temp) \) |
| 3.   | Both parties select a private number, which is not shared over the network ever (e.g. person 1 chooses a, and person 2 chooses b) |
| * NOTE (private numbers are chosen randomly) | |
| 4.   | The receiver calculates the hash value again using steps 2.1 to 2.3 |
| 5.   | Now generate the key in a fashion that Person1Key = \( G' \mod P \) and Person2Key = \( G' \mod P \) |
| a)   | Step 5.1. Person1Key and Person2Key value convert to binary |
| 5.2  | \( l = Calculate the number of 1's in Person1Key and same for Person2Key \) (e.g. \( Temp1 = 3 \)) |
| 5.3  | Person1Key = \( (Person1Key \mod Temp1) \) and Person2Key = \( (Person2Key \mod Temp1) \) |
| 6.   | exchange (Person1Key with person2 and Person2Key with person1) |
| 7.   | Received keys are (person1 = Person2Key) and (person2 = Person1Key) |
| 8.   | The receiver calculates the hash value again using steps 5.1 to 5.3 |
| 9.   | Now use the obtain value to generate the official key |
| 10.  | \( Key_a = Person2Key^a \mod P \) and \( Key_b = Person1Key^b \mod P \) |
| 11.  | Therefore \( Key_a = Key_b \) |
| 12.  | Both parties use the generated key for the encryption and decryption |
4: Person B will calculate the hash value of both P and G using the same process from steps a to g.

5: Person A calculates \( P_1 = G^A \mod P \).

6: Then takes hash of \( P_1 \) using steps a to g.

7: Person B calculates \( P_2 = G^B \mod P \).

8: Then takes hash of \( P_2 \) using steps a to g.

9: Both parties than pass these values to each other.

10: After receiving these values at appropriate place, the hash is again calculated.

11: Person A then calculates \( \text{Key}_A = P_2^A \mod P \).

12: Person B then calculates \( \text{Key}_B = P_2^B \mod P \).

**Phase 2: encryption algorithm**

In Figure 5, the proposed encryption and decryption approach is given. The plaintext is first converted into ASCII decimal values and then converted into binary values. The key is also converted into binary. After obtaining the binary values of both plaintext and key, they are XNORed. Since the key is never able to be known by the attacker, the XNORed will take the plaintext complex and unable to be read. After the XNORed is performed, obtain unreadable data are shifted once toward the left side. This will generate a new unreadable text. Then 1’s complement is performed on the obtained text. The final stage is dividing the temporary ciphertext into two subsets; these subsets swap their positions with each other. The final obtain ciphertext is then passed converted back into ASCII values and passed to the destination.

In the decryption phase, the ciphertext is converted into binary. The obtained data are used, where the text is divided into two subsets. These subsets’ positions are swapped with each other. Then one’s complement is taken of this text to process the decryption. The text obtains from the last step is then shifted once toward the right bit by bit. This text will then be XNORed with the key, which was generated through the Diffie–Hellman. After all this process the original message is generated back again. Using Diffie–Hellman for the key generation, it is not possible to break this cipher. Using multiple operations, the cipher is built more secure.

Since the plaintext is XNORed with the key, therefore it is not able to break until the key is known, and the key will not be known unless the secret variable of either of the parties is known, which is not possible since it is never sent over the network.

Table 6 describes the working of the proposed algorithm in steps. The step is discussed in detail underneath. First, the key is generated using Diffie–Hellman and converted into binary. After the key generation, encryption is performed.

**Encryption process**

**Step 1.** Any plaintext is decided which the sender wants to send to the receiver end.

**Step 2.** After the plaintext is decided, each letter of the plaintext is converted into an ASCII decimal value (e.g. \( a = '97' \)). After the decimal value is calculated, this specific value is then converted into an 8-bit binary (e.g. “97” = 1100001).

**Step 3.** The binary value obtains in step 2 is than XNORed with the same bit key length, generated from Diffie–Hellman (e.g. “10 01 = 00”).

**Step 4.** Results attain from step 3 is shifted on the left side once (e.g. 101 = 011)

**Step 5.** One’s complement is taken of output obtained from step 4.

**Step 6.** The above-generated output is divided into two subsets (e.g. 1011 = 10 and 11). These subsets then change places as (e.g. 1110). This step changes the binary overall, making the algorithm difficult to break.

**Step 7.** After all the encryption steps are performed, there last obtain value in step 6 is converted into ASCII decimal and then into the alphabetic letter.

Table 6 also discusses the decryption process of the proposed algorithm. Decryption is the inverse of encryption steps. The decryption process steps are discussed below in detail.

![Figure 5. Encryption and decryption approach.](image)
Table 6. Secure efficient encryption algorithm (SEEA).

| Encryption: |
|---|
| **INPUT:** |
| 1. Key Generated from Modified Diffie–Hellman |
| 2. Plaintext (already converted in binary values) |
| 3. n = length of the data/letter |
| **OUTPUT:** ciphertext (CT) |

**Encryption:**

\[
\text{temp}_4 = \text{plaintext} \\
1: \text{if key} == \text{available then} \\
2: \text{for each letter 't' in array do} \\
3: \text{if t is part of array then} \\
4: \text{temp}_1 = \text{XNOR(key, plaintext)} \\
5: \text{temp}_2 = \text{left shift each bit once in (temp}_1) \\
6: \text{temp}_3 = \text{take1's complement of(temp}_2) \\
7: \text{temp}_4 = \text{divide data into two sets in (temp}_4) \text{and store as: set1 = } \frac{1}{2^h} \text{ton and } \text{set2 = } \frac{1}{2^h} \text{ton} \\
8: \text{Ciphertext} = \text{swap (Set1, Set2)} \\
9: \text{end if} \\
10: \text{end for} \\
11: \text{else} \\
12: \text{generate key first, then go to step 1} \\
13: \text{end}
\]

**Decryption:**

Obtain ciphertext and then perform the following:

**INPUT:**

1. Key Generated from Modified Diffie–Hellman |
2. Ciphertext (already converted in binary values) |
3. n = length of the data/letter |

**OUTPUT:** plaintext (PT)

\[
\text{temp}_4 = \text{Ciphertext} \\
1: \text{if key} == \text{available then} \\
2: \text{for each letter 't' in array do} \\
3: \text{if t is part of array then} \\
4: \text{temp}_1 = \text{XNOR(key, plaintext)} \\
5: \text{temp}_2 = \text{divide data into two sets in (temp}_4) \text{and store as: set1 = } \frac{1}{2^h} \text{ton and } \text{set2 = } \frac{1}{2^h} \text{ton} \\
6: \text{temp}_3 = \text{take1's complement of(temp}_2) \\
7: \text{temp} = \text{left shift each bit once in (temp}_1) \\
8: \text{Plaintext} = \text{XNOR(key, plaintext)} \\
9: \text{end if} \\
10: \text{end for} \\
11: \text{else} \\
12: \text{generate key first, then go to step 1} \\
13: \text{end}
\]

**Decryption process**

Step 1. The receiver obtains ciphertext, which held not any meaning.

Step 2. After receiving the ciphertext, each letter of the ciphertext is converted into an ASCII decimal value. After the decimal value is calculated, this value is then converted into 8-bit binary

Step 3. The results from step 2 are then divided into two subsets (e.g. 1110 = 11 and 10).

Step 4. These subsets change their places (e.g. 11 and 10 = 1011).

Step 5. One’s complement is taken of results gained from step 4.

Step 6. This output is then shifted toward right once.

Step 7. This result generated in step 6 is XNORed with the private key of the receiver side.

Step 8. After all the decryption steps are performed, the value is then converted into ASCII decimal and then into character (letters).

SEEA is not a heavy encryption algorithm since the encryption steps are kept simple yet the hashing in key generation makes SEEA difficult to break. Hashing appends the data so that the original length of any specific data becomes unknown and make it difficult for the attacker to know the original data. SEEA will reduce have less computational and response time because of its simplicity.

**Phase 3: LEACH**

LEACH for WSNs is introduced to provide secure data routing between two parties. LEACH delivers secure communication between sensor nodes and protect it from various attacks. The working of SEEA on LEACH is given in Table 7. This research approach uses the LEACH protocol for routing. The working of the whole approach is given in Table 7. LEACH protocol uses clustering approach, where clusters heads are selected randomly. After the selection of the cluster head, the communication starts. Multiple steps are discussed in Table 7. These steps show the path of how the communication is built between two nodes using the LEACH protocol and are discussed in detail in the following.

Step 1. The sender sends data or key parameters to its closest cluster head.

Step 2. This sender cluster head sends the data to sink node

Step 3. The sink node route the data to the receiver cluster head

Step 4. This receiver cluster head then forward the data to the receiver node.

The steps 1 to 4 are repeated accordingly. The same process is used when is key is being generated between two nodes and when the communication between two nodes is being formed. An overview of LEACH working is also given in Figure 6.

Step 1. Sender node selects \( A = (1 - n, n = \text{prime number}) \).

Step 2. Receiver node selects \( B = (1 - n, n = \text{prime number}) \).

Step 3. Sender node selects two random number \( P \) and \( G \) and calculates the hash of both values \( h([P] \mod h(P)) \) and \( h(G) \); here \( P \) and \( G \) are random prime
Table 7. Secure efficient encryption algorithm.

| N1 | CHN1 | SN | CNN2 | N2 |
|----|------|----|------|----|
| 1) Select (A = (1 - n, n = prime number)) |
| 3) \( h [P || h(P) & h (G || h(G))] \) |
| 4) \( h [IDN_1, P || h(P) || IDN_1] & (G || h(G) || IDN_1)] \) |
| 5) \( h [IDCHN_1 || (P || h(P)) || IDCHN_1] \) |
| 6) \( h [IDSN || (P; h(P)) || IDCHN_1] \) |
| 7) \( h [IDCHN_2 || (P || h(P)) || IDCHN_2] \) |
| 8) \( h [K2_{temp} || IDN_2 || h (K2_{temp}) = G^B \mod p || IDN_2] \) |
| 9) \( h [K2_{temp} || IDCHN_2 || h (K2_{temp}) = G^B \mod p || IDCHN_2] \) |
| 10) \( h [K1_{temp} || IDSN || h (K1_{temp}) = G^A \mod p || IDSN] \) |
| 11) \( h [K1_{temp} || IDCHN_2 || h (K1_{temp}) = G^A \mod p || IDCHN_2] \) |
| 12) \( h [K1_{temp} || IDN_1 || h (K1_{temp}) = G^A \mod p || IDN_1] \) |
| 13) \( h [K1_{temp} || IDN_1 || h (K1_{temp}) = G^A \mod p || IDN_1] \) |
| 14) \( h [K1_{temp} || IDSN || h (K1_{temp}) = G^A \mod p || IDSN] \) |
| 15) \( h [K1_{temp} || IDCHN_2 || h (K1_{temp}) = G^A \mod p || IDCHN_2] \) |
| 16) Calculate (\( K_{N1} = K2_{temp} \cdot A \mod p \)) |
| 17) Calculate (\( K_{N2} = K1_{temp} \cdot B \mod p \)) |

12) Calculate (\( K_{N1} = K2_{temp} \cdot A \mod p \))

18) Decide (plaintext to be sent)
19) Convert (Plain text to ASCII)
20) Convert (Plain text to Binary)

(Key is Generated) IF (Key is Generated)

(continued)

21) \( \text{tempTxt} = \text{Text} \oplus \text{Key}_{N1} \)
22) \( \text{tempTxt}1 = \lll 1 \) (tempTxt)
23) \( \text{tempTxt}2 = -(\text{tempTxt}1) \)
24) (plaintext/2)
Table 7. Continued

| N1 | CHN1 | SN | CNN2 | N2 |
|----|------|----|------|----|
| 25) Subset1 <-> Subset2 | 26) Output from step 25, covertToASCII | 27) Generated Encrypted Plaintext | 28) [IDSN || Encry (ciphertext || h(IDN))]| |
| 29) [IDCHN1 || Encry (ciphertext || h(IDCHN1))] | 30) [IDSN || Encry (ciphertext || h(IDSN))] | 31) [IDCHN2 || Encry (ciphertext || h(IDCHN2))] |

(Generate key first, then go to step 21)

Figure STYLEREF 1

VI. SEQ Figure 1 ARABIC 1

LEACH Overview

Else

(Generate key first, then go to step 21)

32) Convert (Ciphertext to ASCII)

33) Convert (Ciphertext To Binary)

34) (ciphertext)/2

35) tempTxt = Subset1 <-> Subset2

36) tempTxt2 = -(tempTxt)

37) tempTxt3 = >>1 (tempTxt2)

38) tempTxt3 ⊕ KeyN2

39) covert to ASCII

40) covert to Character

41) Plaintext obtained

Ali et al. 13
numbers and \( h \) for generating a hash value of a given variable.

**Step 4.** Sender node sends the hash of \( P, G \), sender node ID to sender node’s cluster head \( h([ID_{NH}, (P \mod p) || h(P) || ID_{NH}] & (G \mod p) || h(G) || ID_{NH}] \), where \( h \) is again used for hashing and \( P \) and \( G \) are randomly generated prime number. Finally, \( ID_{NH} \) is the ID of the sender node.

**Step 5.** Sender node cluster head sends the hash of \( P, G \), sender node cluster head ID to sink node \( h([ID_{CHNH}, (P \mod p) || h(P) || ID_{NH}] & (G \mod p) || h(G) || ID_{CHNH}] \), where \( h \) is again used for hashing and \( P \) and \( G \) are randomly generated prime numbers. Finally, \( ID_{CHNH} \) is the ID of the sender node cluster head.

**Step 6.** Sink node sends the hash of \( P, G \), sink node ID to receiver node cluster head \( h([ID_{SN} || (P \mod p) || h(P) || ID_{SN}] & (G \mod p) || h(G) || ID_{SN}] \); here \( h \) and \( P \) and \( G \) are the same as mentioned in the above steps. Finally, \( ID_{SN} \) is the ID of the sink node.

**Step 7.** Receiver node cluster head sends the hash of \( P, G \), receiver node cluster head ID to receiver node \( h([ID_{CHrNH}, (P \mod p) || h(P) || ID_{CHrNH}] & (G \mod p) || h(G) || ID_{CHrNH}] \); here, \( h \) and \( P \) and \( G \) are the same as mentioned in the above steps. Finally, \( ID_{CHrNH} \) is the ID of the receiver node cluster head.

**Step 8.** Receiver node sends the hash of \( (K_{temp} = G^B \mod p) \) receiver node ID to receiver node cluster head \( h([K_{temp} \mod p] || ID_{NH2}] ) \), where \( h \) is used for generating hash value. \( K_{temp} \) is the temp value which will be later used for key generation. \( B \) is the secret value of the receiver node. Finally, \( ID_{NH2} \) is receiver node ID.

**Step 9.** Receiver node cluster head sends the hash \( K_{temp} = G^B \mod p \) receiver node ID to sink node \( h([K_{temp} \mod p] || (h(K_{temp} = G^B \mod p) || ID_{CHrNH}]) \), where \( h \) is used for generating hash value. \( K_{temp} \) is the temp value which will be later used for key generation. \( B \) is the secret value of the receiver node. Finally, \( ID_{CHrNH} \) is receiver node cluster head ID.

**Step 10.** Sink node sends the hash of \( K_{temp} = G^B \mod p \) and sink node ID to sender node cluster head \( h([K_{temp} || ID_{SN}] || h(K_{temp} = G^B \mod p) || ID_{SN}] \); here all variables such as \( h \), \( K_{temp} \) and \( B \) are the same as above, and \( ID_{SN} \) is the ID of the sink node.

**Step 11.** Sender node cluster head sends the hash of \( h(K_{temp} = G^B \mod p) \), and sender node cluster head ID to sender node \( h([K_{temp} || ID_{CHrNH}] || h(K_{temp} = G^B \mod p) || ID_{CHrNH}] \); here, all variables such as \( h \), \( K_{temp} \) and \( B \) are the same as above, and \( ID_{CHrNH} \) is the ID of the cluster head of the sender node.

**Step 12.** Sender node sends the hash of \( (K_{temp} = G^A \mod p) \) sender node ID to sender node cluster head \( h([K_{temp} || ID_{CHNH}] || h(K_{temp} = G^A \mod p) || ID_{CHNH}] \), where variable \( h \) is then used to generate a hash value and \( K_{temp} \) is the temporary value which is later used for key generation. \( A \) is the secret value of the sender node. Finally, \( ID_{CHNH} \) is sender node cluster head.

**Step 13.** Sender node cluster head sends the hash \( K_{temp} = G^A \mod p \), sender node cluster ID to sink node \( h([K_{temp} || ID_{CHNH}] || h(K_{temp} = G^A \mod p) || ID_{CHNH}] \); here all variables such as \( h \), \( K_{temp} \), and \( A \) are same as above, and \( ID_{CHNH} \) is the ID of the sender node cluster head.

**Step 14.** Sink node sends the hash of \( K_{temp} = G^B \mod p \) and Sink node ID to receiver node cluster head \( h([K_{temp} || ID_{SN}] || h(K_{temp} = G^B \mod p) || ID_{SN}] \); here all variables such as \( h \), \( K_{temp} \), and \( A \) are same as above, and \( ID_{SN} \) is the ID of the sink node.

**Step 15.** Receiver node cluster head sends a hash of \( h(K_{temp} = G^B \mod p) \), and receiver node cluster head ID to receiver node \( h([K_{temp} || ID_{CHrNH}] || h(K_{temp} = G^B \mod p) || ID_{CHrNH}] \), where all variables such as \( h \), \( K_{temp} \), and \( A \) are same as above, and \( ID_{CHrNH} \) is the ID of the cluster head of the receiver node.

**Step 16.** Sender node calculates its key by \( K_{NH1} = K_{temp} \mod p \); here \( K_{NH1} \) is the sender node key. \( K_{temp} \) is the temp value generated from the above steps. \( A \) and \( P \) are the same as mentioned in the above steps.

**Step 17.** Receiver node calculates its key by \( K_{NH2} = K_{temp} \mod p \); here \( K_{NH2} \) is the receiver node key. \( K_{temp} \) is the temp value generated from the above steps. \( B \) and \( P \) are the same as mentioned in the above steps.

**Step 18.** Sender node selects any plaintext for sending.

**Step 19.** Sender node then converts that plaintext into ASCII value

**Step 20.** Then that ASCII value to binary.
If Key is generated

**Step 21.** Sender node then uses the key generated in step 16 and XNOR it with the binary value in step 20.

**Step 22.** This temporary ciphertext is then left shifted by 1 bit.

**Step 23.** One’s complement of ciphertext in step 22 is taken.

**Step 24.** Text in step 23 is divided into two subsets and then the places of these subsets are swapped.

**Step 25.** After performing all the encryption steps, the obtain binary is then converted into ASCII value.

**Step 26.** Sender node then converts that text into character and this is the finally obtained ciphertext.

**ELSE (if the key not generated then generate key first then go to step 22)**

**Step 27.** Sender node then sends the encrypted text and sender node ID to the sender node cluster head. 

\[ [ID_{N1} \mid Encry(ciphertext) \mid h(ID_{N1})]; \] here \( h \) is the variable used for hashing. Encry is used to mention that the text is encrypted. \( ID_{N1} \) is the ID of the sender node.

**Step 28.** Sender node cluster head sends encrypted text and sender node cluster node ID to sink node. 

\[ [ID_{CH_{N1}} \mid Encry(ciphertext) \mid h(ID_{CH_{N1}})]; \] here \( h \) is the variable used for hashing. Encry is used to mention that the text is encrypted. \( ID_{CH_{N1}} \) is the ID of the cluster head of the sender node.

**Step 29.** Sink node then sends that encrypted text and sink node ID receiver node cluster head. 

\[ [ID_{SN} \mid Encry(ciphertext) \mid h(ID_{SN})]; \] here \( h \) is the variable used for hashing. Encry is used to mention that the text is encrypted. \( ID_{SN} \) is the ID of the sink node.

**Step 30.** The receiver node cluster head then sends the encrypted text and receiver node cluster head ID to the receiver node. 

\[ [ID_{CH_{SN}} \mid Encry(ciphertext) \mid h(ID_{CH_{SN}})]; \] here \( h \) is the variable used for hashing. Encry is used to mention that the text is encrypted. \( ID_{CH_{SN}} \) is the ID cluster head of the receiver node.

**Step 31.** After the receiver node had obtained the encrypted text, it then redoos all steps 19–28 in backward positions.

### Experimental results and discussion

The results are measured based on multiple parameters, such as encryption time, decryption time, response time, and computational cost.

### Simulation setup

The proposed approach uses the LEACH protocol for routing as described in section “Proposed scheme.” It is workable for 5–100 nodes and can run 10 or more rounds in the LEACH protocol. The simulation area on which algorithm was tested is \( 100 \times 100 \). MATLAB is used for implementation. A testbed setup is given in Table 8. An overview of this setup is given in Table 9.

### Setup /environment phase

- The platform used is MATLAB 2013a.
- The proposed approach uses the LEACH protocol for routing.
- The simulation area on which the algorithm is tested is \( 100 \times 100 \).
- The number of nodes used is at most 100 nodes and at least 5 nodes.
- In terms of key, the size can be \( 2^{10} \) maximum and \( 2^{5} \) minima.
- In terms of plaintext, the size can be \( 2^{10} \) maximum and \( 2^{5} \) minima.

### Computational time

The parameter calculates the amount of time an algorithm takes on a specific amount of data. The results of the proposed approach are compared with the research results in the study by Singh et al.\(^ {14} \) Table 2 in the study by Singh et al.\(^ {14} \) shows the performance of their proposed scheme. Where the key generation, encryption, and decryption are performed on 10-byte length of data where the key varies in terms of different ECC values generated. Our proposed approach is tested on three different data lengths, and the key size is the same, yet secure at the same time. The proposed approach updates the key at every round and performs...
better than the research approach in the study by Singh et al. Since the key is generated through the SMDH approach, it is highly difficult to break because the private key of both parties is never passed over the network and random numbers are used to generate common parameters for the public key. Table 10 explains the length of the data versus the amount of time taken to generate key, encryption, and decryption. In Figure 7, the computational time of the SEEA algorithm is given using the data in Table 10.

**Key generation time**

This parameter calculates the time required to generate a key for each round. In Figure 8, key generation time is calculated for different lengths of data given in Table 10. The time is measured in nanoseconds. Diffie–Hellman is used for the unique key generation. Different amount of key length is tested in terms of the amount of time it takes for different lengths of data. The key generation time for data length 10 is about 37,333 nanoseconds, which is suitable for the WSN environment.

**Encryption time**

This parameter calculates the time required for the execution of the proposed methodology of encryption and decryption. The algorithm that takes less encryption time is considered efficient. This proposed encryption algorithm is tested for different lengths of data in terms of the amount of time it takes for each data. Figure 9 discusses the multiple data length versus the amount it takes for them to encrypt. If the data length is 15 bytes, the proposed encryption scheme will take about 59,000 ns to encrypt. 59,000 is 5.9E–5 in seconds, which is very faster. The proposed approach takes very little time to encrypt; therefore, this approach can be considered a better approach for the WSN environment.

| Parameter       | Ciphertext size (byte) | Computational time (ns) | Decryption time (ns) | No. of private key | Encrypted message size (byte) |
|-----------------|------------------------|-------------------------|----------------------|-------------------|-------------------------------|
| Networking      |                        |                         |                      |                   |                               |
| Hello World     | 10                     | Same as message size    | 26,332               | 8                 | 37,333                        |
| My name is      | 65                     | Same as above           | 171,590              | Same as above     | 428,825                       |
| Shahwar Ali, and | 11                     | Same as above           | 167,230              | Same as above     | 428,825                       |
| I am a student  | 65                     | Same as above           | 167,230              | Same as above     | 428,825                       |

*Figure 7. Computational time versus byte of data.*
Decryption time

This parameter calculates the time required for a message to generate plaintext from the obtained ciphertext. This proposed algorithm is tested for different lengths of data in terms of the amount of time it takes for each data to decrypt if the key is known. Figure 10 discusses the multiple data length versus the amount it takes for them to encrypt. If the data length is 45 bytes, the proposed scheme will take about 122,990 ns to decrypt. The time is possible only if the key is known, without knowing the key the decryption would not be possible.

Figures 9 and 13 show the encryption and decryption time of the proposed approach. As the amount of the increase in terms of bytes, encryption, and decryption time increases as well. Proposed approaches take less time as compared to the approach given in Singh et al. \(^\text{14}\) The proposed encryption methodology provides data security, has less encryption time and decryption as shown in Figure 10.

Key generation time comparison

The results in the study by Singh et al. \(^\text{14}\) are compared with the proposed approach on behalf of one string of length 10. The word “Networking” is testing both the existing algorithm and the proposed algorithm. The proposed approach takes less time in key generation, encryption, and decryption compared to the existing approach. Figure 11 shows an overview of the existing approaches versus the key generation approach used in the proposed approach. This figure clearly shows that Diffie–Hellman works faster than the key generation approach in the study by Singh et al. \(^\text{14}\) Since our approach takes less time, therefore, could be considered a better approach than the approach discussed in the study by Singh et al. \(^\text{14}\) As shown in the figure, ECC-109 in the study by Singh et al. \(^\text{14}\) takes 8,975,762 ns and Diffie–Hellman takes 37,333 ns, which is far less than the approach used in the study by Singh et al. The next values, however, are assumed by calculating the percentage of data that are increasing significantly. The equation used to calculate the percentage between the generated data and the existing data as given in equation (1). Where \(V_1\) is the value of the proposed algorithm result, which is in this case 37,333, and \(V_2\) is the other value, such as 7,029,520. To calculate the new value with the percentage, equation (2) is used. In equation (2), \(V_2\) is the existing algorithm values, and the percentage is the value obtained from equation (1). After using these equations, new assumption values are calculated. After the calculation, it can be clearly seen that the proposed algorithm will work better even if the data are increasing significantly.
**Percentage calculation**

\[
\frac{|V_1 - V_2|}{\frac{V_1 + V_2}{2}} \times 100
\]  
(1)

**New value calculation**

\[
V_2 \times (1 + \text{Percentage})
\]  
(2)

**Encryption time comparison**

The proposed encryption approach is compared with the encryption algorithm given in the study by Singh et al.\textsuperscript{14} The word “Networking” is testing both the existing algorithm and the proposed algorithm. Figure 12 shows the proposed approach works far better than the approach in the study by Singh et al.\textsuperscript{14} The given time in Figure 11 is only possible if the key is known to the receiver; otherwise, the decryption process will not be possible. If the key is known, the proposed approach takes 25,853 ns to decrypt, whereas the existing approach in Singh et al.\textsuperscript{14} takes more than 1,244,000 ns. The next values, however, are assumed by calculating the percentage of data that are increasing significantly. The equation used to calculate the percentage between the generated data and the existing data is given in equation (1). Where V1 is the value of the proposed algorithm result, which is in this case 16,450, and V2 is the other value, such as 1,793,860. To calculate the new value with the percentage, equation (2) is used. In equation (2), V2 is the existing algorithm values, and the percentage is the value obtained from equation (1). After using these equations, new assumption values are calculated. After the calculation, it can be clearly seen that the proposed algorithm will work better even if the data are increasing significantly.

**Decryption time comparison**

The proposed approach is compared with the algorithm given in the study by Singh et al.\textsuperscript{14} The word “Networking” is testing both the existing algorithm and the proposed algorithm. Figure 13 shows the proposed approach works far better in terms of decryption time than the approach in the study by Singh et al.\textsuperscript{14} The next values, however, are assumed by calculating the percentage of data that are increasing significantly. The equation used to calculate the percentage between the generated data and the existing data is given in equation (1). Where V1 is the value of the proposed algorithm result, which is in this case 25,853, and V2 is the other value, such as 2312532. To calculate the new value with the percentage, equation (2) is used. In equation (2), V2 is the existing algorithm values, and the percentage is the value obtained from equation (1). After using these equations, new assumption values are calculated. After the calculation, it can be clearly seen that the proposed algorithm will work better even if the data are increasing significantly.

**Response time**

This parameter is used to calculate the time that is taken to react to a given stimulus or event. The response is measured to check how much time does the algorithm takes to execute. In Figure 14, we gave the response time of the proposed approach. As the amount of data increases the response time increases as well. This result showed that the proposed scheme takes less response time for different lengths of data for different rounds, as claimed above in section “Decryption time.” The results are measured in seconds. The results in Figure 15 are calculated in terms of the amount of data versus the number of rounds. As it can be viewed that as the number of rounds increases, the time decreases because, in the first round, the data are being gathered which takes up a few times.

In round 1, the time is less than 3 s for the highest byte of data, but as the number of rounds increases the
time is about less than 1 s for the highest byte of data. The reason for the decrement in time is that the calculations of the whole algorithm get less as the number of round increases and fewer calculations will result in less response time.

SMDH versus Diffie–Hellman

SMDH is a modified version of Diffie–Hellman. Diffie–Hellman is highly vulnerable to MIM attack because this approach shares values over an unauthorized network. the values which are shared over an unreliable network can be accessed by an attacker and the values, therefore, can be modified. To prevent this scenario, SMDH is introduced. SMDH uses hashing approach to produce a hash of each value that will be shared over the network as discussed under section “Proposed scheme.” Hashing helps prevent MIM, which solves the problem in Diffie–Hellman. Table 11 discusses attack that does and does not occur in both Diffie–Hellman and SMDH.

Security and attacks analysis

There are multiple measurements used to check the security of algorithm against multiple attacks. When working with cryptography, multiple attacks can occur through on the network, which needs to be taken care when providing a secure cryptography approach.

Plaintext attack

This attack occurs when the attacker can access both the plaintext and the ciphertext. This attack is considered a very basic attack on cryptography algorithm. When the sender sends data for encryption, the attacker can capture chunks of that plaintext. The key is sent over the secure channel, so it is never known to the attacker. With the help of some known cipher and plaintext, the attacker tries to generate the method used for encryption, which is later used for further decryption of ciphertext. Proposed approach does not send plaintext over the network at all, only ciphertext is sent. Since the key is also never sent over the network, it becomes very difficult for this attack to occur. If the attacker can capture some chunk of ciphertext, it will still be impossible for this attack to occur since the key is updated at each round and the ciphertext changes at each round. An overview of how this attack is not possible while following the proposed approach is given in Figure 15.

COA

In cryptography, a COA or known-ciphertext attack is an attack model for cryptanalysis where the attacker is presumed to have access only to a set of ciphertexts. The attack is completely successful if the corresponding plaintexts can be deduced (extracted) or, even better, the key.

It is very difficult for this attack to occur, because even though ciphertext is passed over the network, but the key is never sent over the network. Proposed approach shares key through common parameters rather than using a secure channel. Since secure channel is not used to pass the key, key will not be known to the attacker which makes it very difficult for the ciphertext to decrypt. Without the key, it is not possible for the ciphertext to be decrypted there is a high chance that this attack will not occur while using the proposed scheme. Figure 16 shows an over of how this attack is not possible on the proposed approach. Analyzing the
ciphertext is possible if key can be recovered. It is very difficult to recover key while using the proposed approach and this makes it very challenging for this attack to occur.

**Related-key attack**

In cryptography, a related-key attack is any form of cryptanalysis where the attacker can observe the operation of a cipher under several different keys whose values are initially unknown, but where some mathematical relationship connecting the keys is known to the attacker. In Figure 17, we show a scenario of how this attack is not possible on proposed approach.

For example, the attacker might know that the last 80 bits of the keys are always the same, even though they do not know, at first, what the bits are. Since proposed approach does not use the same key at each round of LEACH protocol, this attack becomes difficult to occur. In case if the key is known to the attacker, that key will only work for one encrypted text, since new key is generated at every round. It is very difficult for the attacker to obtain the key because only specific parameters are used to generate unique and the private parameters are always kept secret.

**MIM attack**

A MIM attack involves three players. There is the victim, the entity with which the victim is trying to communicate, and the “man in the middle,” who’s interrupting the victim’s communications. In Figure 18, a scenario of how this attack is not possible on proposed algorithm is given.

Diffie–Hellman is highly vulnerable to MIM. This attack will lower the security level by accessing all the private parameter values. To secure proposed approach from this attack, hashing is used in Diffie–Hellman. After modifying Diffie–Hellman using hash function, this proposed is secure from MIM attack. Only two communicating parties can generate the correct hash value, which is why this attack becomes meaningless in proposed approach.

Further man in the middle attack is possible if the attacker can hear the secret conversation between the two communicating parties. In proposed scenario, if the attacker can capture the packets being sent between the two parties, the obtained packets will be encrypted. The encrypted data can only be decrypted if the key and the encryption algorithm is known. Since private and public key concepts are used for the generation of keys, it is very difficult to know the key because the private key is never sent over the network. If the key is not known, it becomes very difficult for an unauthentic person to recover the data. Symmetric key is share over the network, where the attacker can obtain the key and use it to recover the data. However, asymmetric approach
does not require the key to be sent over the network, which makes proposed approach more secure against man in the middle attack.

Conclusion

A WSN is a spread network and it contains a large amount of circulated, self-focused, tiny, low motorized devices called sensor nodes. Even though the sensor network is commonly being used, they are very complex due to the limited amount of energy and memory they can use. With multiple other issues, data security is one of the major issues in WSNs, since the data traveling over the network is never safe due to the reason that multiple attackers can access the data. The data need to be protected from the attacker, which is why the data are encrypted to turn it into a form that is not readable. Multiple approaches have been used to encrypt the data for security, but due to some flaws, these approaches are not considered suitable for WSNs. In the study by Elhoseny et al., another approach is given where the author uses the ECC, node ID, and distance between the node to generate the key and then this key is used for encryption and decryption. This method is helpless to brute-force attack. The usage of PRNG in this approach can help the intruder to use the short period of pseudo-random number generator (PRNG) to generate private keys. In the study by Ullah et al., another technique of data security for WSNs is given. The following approach uses HECC for key generation and AES is used for the encryption of the data. The high computational time of AES is a major flaw with additional flaw usage of random number generator, there exist methods that can be used to decrypt a random number generator and usage of AES makes that biclique attacks occur in AES, as proved in the study by Bogdanov et al. To overcome the drawbacks mentioned above, a secure light data encryption approach is proposed. This approach will have less computational time. Since proposed methodology has less computational and response time, it can be considered the best fit for WSN security. The proposed approach also prevents plaintext, ciphertext-only, related-key, and man in the middle attacks.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This project was funded by the Deanship of Scientific Research (DSR), King Abdul-Aziz University, Jeddah, funded this project under grant no. (D-370-611-1441). The authors, therefore, gratefully acknowledge DSR technical and financial support.

ORCID iD

Imran Khan https://orcid.org/0000-0003-1319-5970

References

1. Prusty AR. The network and security analysis for wireless sensor network: a survey. Int J Comput Sci Inf Technol 2012; 3(3): 4028–4037.
2. Akyildiz IF, Su W, Sankarasubramaniam Y, et al. Wireless sensor networks: a survey. Comput Netw 2002; 38(4): 393–422.
3. Elhoseny M, Elminir H, Riad A, et al. A secure data routing schema for WSN using elliptic curve cryptography and homomorphic encryption. J King Saud Univ–Comput Inf Sci 2016; 28(3): 262–275.
4. Mahmood K, Arshad J and Chaudhry SA. An enhanced anonymous identity-based key agreement protocol for smart grid advanced metering infrastructure. Int J Commun Syst 2019; 32(16): e4137.
5. Ullah I, ul Amin N, Iqbal J, et al. An efficient secure protocol for wireless sensor networks based on hybrid approach. IJCSNS 2018; 18(6): 59.
6. Tiwari HD and Kim JH. Novel method for DNA-based elliptic curve cryptography for IoT devices. ETRI J 2018; 40(3): 396–409.
7. Wang W, Si M, Pang Y, et al. An encryption algorithm based on combined chaos in body area networks. Comput Electr Eng 2018; 65: 282–291.
8. Ahmad M, Al Solami E, Wang X-Y, et al. Cryptanalysis of an image encryption algorithm based on combined chaos for a BAN system, and improved scheme using SHA-512 and hyperchaos. Symmetry 2018; 10(7): 266.
9. Chaudhry SA, Shon T, Al-Turjuman F, et al. Correcting design flaws: an improved and cloud assisted key agreement scheme in cyber physical systems. Comput Commun 2020; 153: 527–537.
10. Mansoor K, Ghani A, Chaudhry SA, et al. Securing IoT based RFID systems: a robust authentication protocol using symmetric cryptography. Sensors 2019; 19(21): 4752.
11. Somsuk K and Sanemueang C. The new modified methodology to solve ECDLP based on brute force attack. In: International conference on computing and information technology, Chiangmai, Thailand, 5–6 July 2018, pp.255–264. Cham: Springer.
12. Santhosh R and Shalini M. Security enhancement using chaotic map and secure encryption transmission for wireless sensor networks. Int J Eng Technol 2017; 9(2): 689–694.
13. Daemen J. Limitations of the Even-Mansour construction. In: International conference on the theory and application of cryptography and information security, Gold Coast, QLD, Australia, 13–16 December 1991, pp.495–498. Berlin: Springer.
14. Singh SR, Khan AK and Singh TS. A new key management scheme for wireless sensor networks using an elliptic curve. Indian J Sci Technol 2017; 10(13): 1–7.
15. Wiener MJ and Zuccherato RJ. Faster attacks on elliptic curve cryptosystems. In: International workshop on selected areas in cryptography, Kingston, ON, Canada, 17–18 August 1998, pp.190–200. Berlin: Springer.

16. Neves S and Tibouchi M. Degenerate curve attacks: extending invalid curve attacks to Edwards curves and other models. IET Inf Secur 2017; 12(3): 217–225.

17. Viswanathan S and Kamann A. Elliptic key cryptography with Beta Gamma functions for secure routing in wireless sensor networks. WIRE Netw 2019; 25: 4903–4914.

18. Ghani A, Mansoor K, Mehmood S, et al. Security and key management in IoT based wireless sensor networks: an authentication protocol using symmetric key. Int J Commun Syst 2019; 32(16): e4139.

19. Reeds J. “Cracking” a random number generator. Cryptologia 1977; 1(1): 20–26.

20. Bogdanov A, Khovratovich D and Rechberger C. Biclique cryptanalysis of the full AES. In: International conference on the theory and application of cryptography and information security, Seoul, Korea, 4–8 December 2011, pp.344–371. Berlin: Springer.

21. Roy S and Khatwani C. Cryptanalysis and improvement as asymmetric encryption padding (OAEP) as standardized and key exchanging protocols. Cryptography 2017; 1(1): 9.

22. Suresh H and Hegadi RS. DCA-SNC: dual cryptosystem architecture for secure network communication. Int J Adv Res Comput Sci Software Eng 2017; 7(1): 198–205.

23. Hussain S and Chaudhry SA. Comments on “Biometrics-based authentication and key exchanging protocols.” Int J Comput Sci Inf Secur 2015; 13(8): 80.

24. Nitaj A, Ariffin MRK, Nassr DI, et al. New attacks on the RSA cryptosystem. In: International conference on cryptography in Africa, Marrakesh, Morocco, 28–30 May 2014, pp.178–198. Cham: Springer.

25. De Weger B. Cryptanalysis of RSA with small prime difference. Appl Algebra Eng Commun Comput 2002; 13(1): 17–28.

26. Roy D and Das P. A modified RSA cryptography algorithm for security enhancement in vehicular ad hoc networks. In: Mandal J, Saha G, Kandar D, et al. (eds) Proceedings of the international conference on computing and communication systems. Singapore: Springer, 2018, pp.641–653.

27. Manger J. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS# 1 v2. 0. In: Annual international cryptology conference, Santa Barbara, CA, 19–23 August 2001, pp.230–238. Berlin: Springer.

28. Lindell Y and Katz J. Introduction to modern cryptography. London: Chapman and Hall/CRC, 2014.

29. Ali Z, Chaudhry SA, Ramzan MS, et al. Securing smart city surveillance: a lightweight authentication mechanism for unmanned vehicles. IEEE Access 2020; 8: 43711–43724.

30. Babu SS and Balasubadra K. Revamping data access privacy preservation method against inside attacks in wireless sensor networks. Chast Comput 2019; 22: 65–75.

31. Haakegaard R and Lang J. The Elliptic Curve Diffie-Hellman (ECDH), http://koclab.cs.ucsb.edu/teaching/ ecc/project/2015Projects/haakegaard + Lang.pdf.

32. Li J. A symmetric cryptography algorithm in wireless sensor network security. Int J Online Eng 2017; 13(11): 102–110.

33. Biryukov A and Khovratovich D. Related-key cryptanalysis of the full AES-192 and AES-256. In: International conference on the theory and application of cryptography and information security, Tokyo, Japan, 6–10 December 2009, pp.1–18. Berlin: Springer.

34. Khan A, Shah SW, Ali A, et al. Secret key encryption model for wireless sensor networks. In: 2017 14th international Bluruhan conference on applied sciences and technology (IBCAST), Islamabad, Pakistan, 10–14 January 2017, pp.809–815. New York: IEEE.

35. Vangala A and Parwekar P. Enhanced encryption model for sensor data in wireless sensor network. In: 2017 20th international symposium on wireless personal multimedia communications (WPMC), Bali, Indonesia, 17–20 December 2017, pp.16–21. New York: IEEE.

36. Albassal EMB and Wahdan A-M. Genetic algorithm cryptanalysis of the basic substitution permutation network. In: 2003 IEEE 46th midwest symposium on circuits and systems, Cairo, 27–30 December 2003, pp.471–475. New York: IEEE.

37. Vangala A and Parwekar P. Encryption model for sensor data in wireless sensor networks. In: Bhatjea V, Nguyen B, Nguyen N, et al. (eds) Information systems design and intelligent applications. Singapore: Springer, 2018, pp.963–970.

38. Kelsey J, Schneier B, Wagner D, et al. Cryptanalytic attacks on pseudorandom number generators. In: International workshop on fast software encryption, Paris, 23–25 March 1998, pp.168–188. Berlin: Springer.

39. Al-Mashhadi HM, Abdul-Wahab HB and Hassan RF. Data security protocol for wireless sensor network using chaotic map. Int J Comput Sci Inf Secur 2015; 13(8): 80.

40. Bayılmış C, Çavuşoğlu Ü, Akgül A, et al. Enhanced secure data transfer for WSN using chaotic-based encryption. Teh Vjesn 2017; 24(4): 1065–1069.

41. Alvarez G, Montoya F, Romera M, et al. Cryptanalysis of dynamic look-up table based chaotic cryptosystems. Phys Lett A 2004; 326(3–4): 211–218.

42. Nidarsh MP and Devi MGP. Chaos based secured communication in energy efficient wireless sensor networks. Chaos 2018; 5(6): 742–747.

43. Sobhy MI and Shehata A-E. Methods of attacking chaotic encryption and countermeasures. In: 2001 IEEE international conference onacoustics, speech, and signal processing, Salt Lake City, UT, 7–11 May 2001, pp.1001–1004. New York: IEEE.

44. Li L, Liu B and Wang H. QTL: a new ultra-lightweight block cipher. Microprocess Microsyst 2016; 45: 45–55.

45. Sadeghi S, Bagheri N and Abdelraheem MA. Cryptanalysis of reduced QTL block cipher. Microprocess Microsyst 2017; 52: 34–48.

46. Patil J, Bansod G and Kant KS. LiCi: a new ultra-lightweight block cipher. In: International conference on emerging trends innovation in ICT (ICEI), Pune, India, 3–5 February 2017, pp.40–45. New York: IEEE.

47. Bogdanov A, et al. PRESENT: an ultra-lightweight block cipher. In: Cryptographic hardware and embedded systems
(CHES 2007), Vienna, 10–13 September 2007, pp.450–466. Berlin: Springer.

48. Maitly S, Sinha K and Sinha BP. An efficient lightweight stream cipher algorithm for wireless networks. In: Wireless communications and networking conference (WCNC), San Francisco, CA, 19–22 March 2017, pp.1–6. New York: IEEE.

49. Li S, Li C, Chen G, et al. A general cryptanalysis of permutation-only multimedia encryption algorithms. IACR Cryptol Eprint Arch Rep 2004; 2004: 374.

50. Solomon M and Elias EP. Data security and privacy in wireless sensor devices. Networks 2018; 5(5).

51. Wang Y and Yang M. Higher order differential cryptanalysis on the SHA-3 cryptographic hash algorithm competition candidates, 2009. https://pdfs.semanticscholar.org/4d10/459de1896b2f392e7e88d9d608129cf9e4d43.pdf?_ga=2.81639319.1678718536.158929567-209426266.1549527293

52. Kocher PC. Cryptanalysis of Diffie-Hellman, RSA, DSS, and other systems using timing attacks (Extended Abstract) (1995). In: Annual international cryptography conference, Santa Barbara, CA, 18–22 August 1996. Berlin: Springer.

53. Praveena A. Achieving data security in wireless sensor networks using ultra encryption standard version #x2014; IV algorithm. In: 2017 international conference on innovations in green energy and healthcare technologies (IGEHT), Coimbatore, India, 16 March 2017, pp.1–5. IEEE.

54. Lasry G, Kopal N and Wacker A. Cryptanalysis of columnar transposition cipher with long keys. Cryptologia 2016; 40(4): 374–398.

55. Li C, Li S, Alvarez G, et al. Cryptanalysis of two chaotic encryption schemes based on circular bit shift and XOR operations. Phys Lett A 2007; 369(1–2): 23–30.

56. Petters J. What is a Brute Force Attack? Definition—Varonis. Inside Out Security, Updated 16 October 2018, https://www.varonis.com/blog/brute-force-attack/ (accessed 21 July 2018).

57. Ananda Krishna B, Madhuri N, Rao MK, et al. Implementation of a novel cryptographic algorithm in wireless sensor networks. In: 2018 conference on signal processing and communication engineering systems (SPACES), Vijayawada, India, 4–5 January 2018, pp.149–153. New York: IEEE.

58. Peng X, Zhang P, Wei H, et al. Known-plaintext attack on optical encryption based on random double phase keys. Opt Lett 2006; 31(8): 1044–1046.

59. Praveena A and Smys S. Efficient cryptographic approach for data security in wireless sensor networks using MES VU. In: 2016 10th international conference on intelligent systems and control (ISCO), Coimbatore, India, 7–8 January 2016, pp.1–6. New York: IEEE.

60. Gao Y, Ao H, Feng Z, et al. Mobile network security and privacy in WSN. Procedia Comput Sci 2018; 129: 324–330.

61. Shruthi GJ. Digital signature based key management protocol for secure data transfer in dynamic wireless sensor networks. In: IEEE international conference on recent trends in electronics, information & communication technology (RETEICT), Bangalore, India, 20–21 May 2016, pp.499–502. New York: IEEE.

62. Irshad A, Usman M, Ashraf Chaudhry S, et al. A provably secure and efficient authenticated key agreement scheme for Energy Internet based Vehicle-to-Grid technology framework. IEEE T Ind Appl. Epub ahead of print 13 January 2020. DOI: 10.1109/TIA.2020.2966160.

63. Ull Hassan M, Chaudhry SA and Irshad A. An improved SIP authenticated key agreement Based on Dongqin et al. Wireless Pers Commun 2020; 110: 2087–2107.

64. Alzahrani BA, Chaudhry SA, Barnawi A, et al. A privacy preserving authentication scheme for roaming in IoT-based wireless mobile networks. Symmetry 2020; 12(2): 287.

65. Gracy PL and Venkatesan D. An honey encryption based efficient security mechanism for wireless sensor networks. Int J Pure Appl Math 2018; 118(20): 3157–3164.

66. Degan S. Cryptanalysis of hash functions, 2012, https://www.esat.kuleuven.be/cosic/publications/thesis-223.pdf

67. Yue T, Wang C and Zhu Z. Hybrid encryption algorithm based on wireless sensor networks. In: 2019 IEEE international conference on mechatronics and automation (ICMA), Tianjin, China, 4–7 August 2019. New York: IEEE.

68. Luo Y, Yao L, Liu J, et al. A block cryptographic algorithm for wireless sensor networks based on hybrid chaotic map. In: 2019 IEEE 21st international conference on high performance computing and communications; IEEE 17th international conference on smart city; IEEE 5th international conference on data science and systems (HPCC/Smart/City/DSS), Zhangjiajie, China, 10–12 August 2019. New York: IEEE.

69. Laoud A, Dahmani A, Hassen HR, et al. A self-managing volatile key scheme for wireless sensor networks. J Amb Intell Hum Comput 2019; 10: 3349–3364.

70. Fotofi R, Firoozi Bari S and Yusefi M. Securing wireless sensor networks against denial-of-sleep attacks using RSA cryptography algorithm and interlock protocol. Int J Commun Syst 2019; 33: e4234.

71. Boubiche DE, Athmani S, Boubiche S, et al. Cybersecurity issues in wireless sensor networks: current challenges and solutions. Wireless Pers Commun. Epub ahead of print 10 March 2020. DOI: 10.1007/s11277-020-02721-5.

72. Pooja and Chauhan RK. Triple phase hybrid cryptography technique in a wireless sensor network. Int J Comput Appl. Epub ahead of print 7 January 2020. DOI: 10.1080/120621x.2019.1710342.

73. Saarinen M-JO. Cryptanalysis of block ciphers based on SHA-1 and MD5. In: International workshop on fast software encryption, Lund, 24–26 February 2003, pp.36–44. Berlin: Springer.

74. Habeeb IJ and Muhajjar RA. Secured wireless sensor network using improved key management. In: Proceedings of the fifth international conference on network, communication and computing, Kyoto, Japan, 17–21 December 2016, pp.302–305. New York: ACM.

75. Kaliski BS and Yin YL. On differential and linear cryptanalysis of the RC5 encryption algorithm. In: Annual international cryptology conference, Santa Barbara, CA, 27–31 August 1995, pp.171–184. Berlin: Springer.
76. Rani A and Kumar S. A low complexity security algorithm for wireless sensor networks. In: *2017 innovations in power and advanced computing technologies (i-PACT)*, Vellore, India, 21–22 April 2017, pp.1–5. New York: IEEE.

77. Tsai K-L, Leu F-Y, Su T-H, et al. A light weight data encryption method for WSN communication. In: *International conference on broadband and wireless computing, communication and applications*, Barcelona, 8–10 November 2017, pp.788–795. Cham: Springer.

78. Wang X, Yin YL and Yu H. Finding collisions in the full SHA-1. In: *Annual international cryptology conference*, Santa Barbara, CA, 14–18 August 2005, pp.17–36. Berlin: Springer.

79. Madhuravani B, Murthy DSR and Raju SV. An improved wireless node neighbor integrity verification and encryption using additive and multiplicative homomorphic model. *J Fundam Appl Sci* 2018; 10(6S): 2911–2932.

80. Biham E. Cryptanalysis of the chaotic-map cryptosystem suggested at EUROCRYPT’91. In: *Workshop on the theory and application of cryptographic techniques*, Brighton, 8–11 April 1991, pp.532–534. Berlin: Springer.

81. Reddy PLK, Reddy BRB and Krishna SR. Secure random bit size encryption algorithm for wireless sensor data transmission. *Int J Sci Res Comput Sci Eng Inform Technol* 2018; 3: 371–380.

82. Coron J-S, Joye M, Naccache D, et al. New attacks on PKCS# 1 v1.5 encryption. In: *International conference on the theory and applications of cryptographic techniques*, Bruges, 14–18 May 2000, pp.369–381. Berlin: Springer.

83. Prakash S and Rajput A. Hybrid cryptography for secure data communication in wireless sensor networks. In: Hu Y-C, Tiwari S, Mishra KK, et al. (eds) *Ambient communications and computer systems*. Singapore: Springer, 2018, pp.589–599.

84. Coron J-S and Joux A. Cryptanalysis of a provably secure cryptographic hash function. *IACR Cryptol Eprint Arch* 2004; 2004: 13.

85. Diffie W and Hellman M. New directions in cryptography. *IEEE T Inform Theory* 1976; 22(6): 644–654.