HCIS | Human-centric Computing and Information Sciences

RESEARCH (Research Manuscript) **Open Access**

Human-centric Computing and Information Sciences (2023) 13:05
DOI: https://doi.org/10.22967/HCIS.2023.13.005
Received: March 22, 2022; Accepted: March 28, 2022; Published: January 30, 2023

A Low Overhead and Scalable Authentication and Encryption Scheme for Medical Wireless Sensor Networks

Harpreet Vohra[1], Mohammad Kamrul Hasan[2,*], Harsh Shukla[1], Ravie Chandren Muniyandi[2], Hesham Alhumyani[3], Mohammed S. Alzaidi[4], Manpreet S Manna[5], Shayla Islam[6,*], and A. K. M. Ahasan Habib[2]

Abstract

This study discusses an integrated authentication and encryption (LoSWIAE) scheme for medical wireless sensor networks (MWSNs). A scalable watermarking-based solution promises amalgamated advantages offered by both encryption and authentication schemes in MWSNs but at much lower costs. Furthermore, unlike encryption, the proposed scheme does not require an encryption–decryption key to be exchanged before communication. LoSWIAE ensures nonrepudiation, confidentiality, and protection against eavesdropping in addition to node and data authentication. The scheme's strength lies in the node's geographical credentials, temporal credentials of the message, and basic information captured by the sensor node. Parameters used to check the scheme's robustness are cracking probability and communication overhead. Analytical results obtained compared to an earlier scheme show the effectiveness of LoSWIAE. It proves to have better efficiency and an improvement of the order of 1,032 at the cost of infinitesimal communication overhead.

Keywords

Lightweight, Node Authentication, Watermarking, Wireless Sensor Network, IoT, Cloud Computing

1. Introduction

Today, few advanced modern technologies like cloud computing and the Internet of Things (IoT) depend on the network and Internet services for communication, making cyber security a thrust area for research. Due to the ease of implementation and innumerable applications wherein wired networking is impossible, the shift to wireless networks seemed more promising, specifically when the number of wireless networks is more. Most of the ultramodern applications, e.g., cellular, adhoc, medical wireless sensor networks (MWSNs), mobile adhoc (MANETs), vehicular adhoc (VANETs), and peer-to-peer networks communicate wirelessly [1–3]. IoT has carved its niche in the field of the medical domain too.

***Corresponding Author:** Mohammad Kamrul Hasan (mkhasan@ukm.edu.my), Shayla Islam (shayla@ucsiuniversity.edu.my)
[1]Electronics and Communication Engineering Department, Thapar Institute of Engineering and Technology, India.
[2]Center from Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM), Selangor, Malaysia.
[3]Department of Computer Engineering, College of Computers and Information Technology, Taif University, Taif, Saudi Arabia.
[4]Department of Electrical Engineering, College of Engineering, Taif University, Taif, Saudi Arabia.
[5]Electrical and Instrumentation Department, Sant Longowal Institute of Engineering and Technology, Longowal, Punjab, India.
[6]Institute of Computer Science and Digital Innovation, UCSI University, Kuala Lumpur, Malaysia.

Various medical devices, healthcare practitioners, and patients communicate using the Internet to realize what is termed the Internet of Medical Things (IoMT). In IoMT, data is shared and saved through the cloud platforms using different gateways (Fig. 1). Cloud computing has emerged as an effective solution for data sharing and other services between numerous technologically diverse systems. However, such innovations have brought in significant challenges related to security and privacy issues [4]. A pacemaker carried by a patient can be one such example prone to an unauthorized attack on a healthcare device.

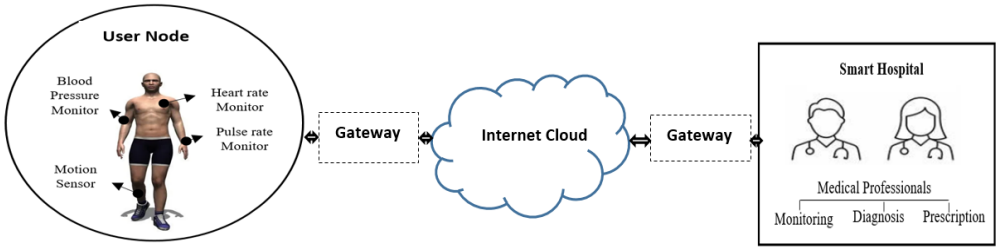


Fig. 1. Conceptual model of IoMT architecture.

The 5G networks, which are considered the backbone of the wireless communication system, are prone to challenges like Flash network traffic specific to the IoT environment, security of encrypted keys shared over the radio transmission, security of applications running on the end users' devices, and so on [5]. Another such grave example, called advanced persistent threat, is a cyberattack done on a specific enterprise for accessing its financial and other secret information [6]. Several studies are being done in the last two to three decades to address various security challenges in wireless communication. The possible solutions explored so far include access control, flow control, security identification and verification, service access control, location security, and so on. Meanwhile, the penetration of blockchain in cloud computing has risen as a solution to various vulnerability and load issues safeguarding the interactions, data storage, and trust involved in applications like healthcare, finance sector, and so on.

Keeping in view the tremendous need for secure systems, an effective encryption and authentication scheme is required to address the concerns discussed above while maintaining a low communication overhead. This paper proposes a low overhead and scalable watermarking-based integrated authentication and encryption (LoSWIAE) technique that provides node and data authentication. It also supports security against various attacks and protects against eavesdropping. The key feature of LoSWIAE which differentiates it from other schemes is that it does not need the exchange of the key between the nodes before the communication unlike other encryption schemes, which require additional overhead for the key distribution mechanisms. Moreover, scalability, the backbone of LoSWIAE, differentiates it from other schemes and provides unparalleled robustness because it is reversible and self-sustainable. Furthermore, it utilizes the decryption key, hidden as a watermark in the communicated message. The node must first authenticate itself to obtain the decryption key. LoSWIAE neither computes the hash digest of the message to be communicated nor does it use cryptographic algorithms—e.g., advanced encryption standard (AES) and elliptic curve cryptography (ECC). Instead, simple single-clock cycle operations (e.g., XOR, shift, insertion, and extraction) are used, thus reducing the computation cost. It maintains the scalability in the length of the message to be communicated, reducing the communication cost. For instance, in a scheme incorporating MD5 [7], a 128-bit hash digest is computed for a 64-bit message leading to an additional 64-bit communication overhead which is not the case with LoSWIAE. Meanwhile, the security, as provided by hashing algorithms (e.g., MD5 against man in the middle attack) is not compromised in LoSWIAE because the message is encrypted using the means of interpolating and watermarking.

The rest of the paper's organization is as follows: Section 2 briefly reviews the existing work; Section 3 delineates the proposed scheme and further illustrates the scheme with the help of an example; and an

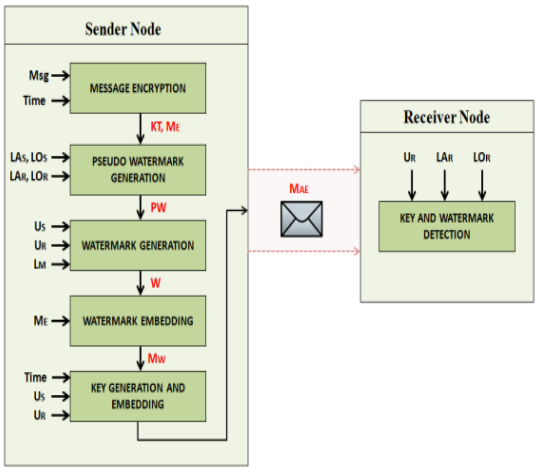
analysis of the performance of the LoSWIAE scheme is presented in Section 4, demonstrating how scalability in the proposed algorithm lowers the cracking probability by multifold. The results show that LoSWIAE provides strong robustness against threats like identity replication attack [8], masquerade attack [8], man-in-the-middle attack (MITM) [9, 10], forward and backward secrecy [11], and so on making it a trustworthy candidate to be employed as a security measure in resource-constrained wireless sensor networks (WSNs). Section 5 provides a comparative analysis of LoSWIAE with a recently published watermark-based node authentication scheme. Section 6 concludes the work and gives an insight into possible future studies.

2. Literature Survey

The rapid growth in the field of WSNs equipped with the power of the Internet has benefited many application areas. The introduction of IoT in domains (i.e., healthcare, industry, and so on) has brought paradigm shifts like IoMT, industrial IoT (IIoT), and so on. The coronavirus disease 2019 (COVID-19) pandemic made people realize the importance of telemedicine even more. Through smart biosensors serving the purpose of real-time health monitoring and the Internet, people like doctors, caregivers, and patients can exchange information in real-time. Likewise, as per the doctor's instructions, automated smart equipment (i.e., pacemaker, insulin delivery systems, and so on) are operated. However, all these benefits have come with the increased susceptibility to unwanted attacks at the sensor locations, cloud, and intelligent equipment. Numerous techniques have been developed to ensure the authenticity of the wireless nodes and data encryption to ensure the safe delivery of the messages. Mendilah et al. [12] proposed a FlexenTech encryption technique using IoT to increase security and decrease the computation time in data transmission by minimizing the computations and number of rounds used to cipher the information. Secure communication between the various MWSN nodes kept at remote locations with compromised power supplies and human intervention is crucial [13–16]. Zhang et al. [17] proposed an entity authentication scheme that incorporates a six-phase node authentication framework and uses the concept of hidden point generator and ECC [18] as its backbone. A dual watermarking frame for content authentication and temporal localization using discrete cosine transform coefficients based on IoT technology for the industrial environment is proposed [19]. A secure sharing of information between spatially different network nodes is ensured with the emerging co-use of blockchain and IoT. An approach proposed by Wang et al. [20] involves pairing a free lightweight blockchain-based certificateless signature scheme to solve the data privacy and security concerns for IoT-based systems. A critical objective while using blockchain is the preservation of authentic digital signatures produced by a node against the forged signatures sent by an eavesdropper in applications (i.e., Bitcoin) [21]. The approach addresses the problems of efficiency improvement for false signature isolation in batches and their identification [ECDSA]. Likewise, for network function virtualization systems, intrusion detection systems (IDS) are being used to stop malicious attacks [22]. Deep neural network-based IDS in IoMT architectures is proposed in [23]. The paper highlights that the secure exchange of the encryption keys among the different systems is still a huge challenge that needs a solution. An emerging solution for the hardware security of the sensor nodes is the use of physically unclonable functions (PUF) [24] with the random characteristic. It is a hardware equivalent to the biometric handprint capable of generating the key on-demand without fixed key storage. A physically unclonable functions and blockchain-based data authentication scheme is proposed in [25], addressing the physical security of nodes and gateway nodes. A detailed analysis of different approaches to address the APT attacks is given in [6]. Algorithms (i.e., AES [26], Secure Hash Algorithm [27], and so on), have proven strong robustness to the network and are found to be highly computationally intensive. Low overhead watermark-based node authentication (LoWaNA) [28] can be used for node authentication in flat architectures and unicast communication. Moreover, LoWaNA provides security against masquerade [29] and replays attacks [7, 30]. However, the scheme incorporates the use of MD5 which makes it computationally intensive.

3. LoSWIAE Algorithm

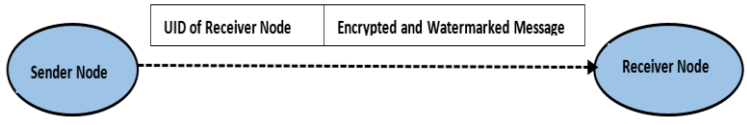
The proposed watermarking-based solution consists of six algorithms, with each algorithm adding a layer of robustness over the previous one. Fig. 2(a) represents the flow model of LoSWIAE. The sender node computes the secured message by executing message encryption, pseudo-watermark generation, watermark generation, watermark embedding, and key generation and embedding. The message further routes along the path to the intended receiver node. The receiver node executes the sixth algorithm key and watermark detection first to authenticate itself and later to deduce the decryption key. The LoSWIAE network model considers unicast, broadcast, and many-to-one modes of communication in WSNs. Many-to-one modes of communication are observed in the case of multiple sensor nodes sending the data to a data aggregation point or the base station. The nodes are assumed to be deployed in a hostile area and are carrying crucial information. Each node's unique identity (UID) number is changed after every predefined time interval by the base station to enhance the security of the nodes in the network. Each node maintains a lookup table with the number of entries equal to the number of nodes in the network. Each entry consists of a four-tuple data consisting of its cluster-ID (Fig. 2(b)), UID number of the node, and the node's location in terms of the geographical latitude and the longitude information.



Notation	Information
LA_s	Latitude of the sender node
LA_R	Latitude of the receiver node
LO_s	Longitude of the sender node
LO_R	Longitude of the receiver node
U_S	Unique identity number of the sender node
U_R	Unique identity number of the receiver node
KT	Encryption key generated
L_M	Length of the message
L_K	Length of string constituting key time
PW	Pseudo watermark generated
W	Watermark generated
M_E	Bit stream generated after encrypting the information with the encryption key
Time	Time at which the message is generated
M_W	Partially Watermarked message
M_{AE}	Bit stream generated after embedding all the watermarked bits i.e. the bit stream to be sent to the receiver node
Key	Key generated using the time at which the message is generated
B,B_S,B_N	Block, Block size, Block Number
U_L	Length of the unique identity number
Count	Variable to store the number of 1's

(a)

(b)



(c)

Fig. 2. (a) Flow of the proposed scheme with notations, (b) unique identifier of sensor node/cluster node, (c) two-tuple packet format for LoSWIAE.

The length of the message transmitted between the sender and receiver nodes depends on the raw information captured by the sensor nodes, the unique identity numbers of the communicating nodes, and the time at which the information is transmitted in a two-tuple format consisting of the UID of the receiver node and the watermarked message (Fig. 2(c)). The two tuples appear segregated from the authentic

nodes in the network. However, the line of division between the two tuples is unclear to an outsider. In the case of hierarchical architecture, the sensor node in a cluster communicates the information to its cluster head. It further processes the information obtained from all the sensor nodes, agglomerates it whenever necessary, and sends it to the next level cluster head (base station) or broadcasts it to all the nodes in the cluster. The process continues until the information reaches the desired user or the base station. In such a case, the UID field of the transmitted packet consists of the identification number of the cluster head. In the case of flat architecture, transmission occurs in a peer-to-peer fashion, i.e., the sensor node communicates either with the nearby sensor nodes.

3.1 Message Encryption

LoSWIAE aims to build robustness in the network by incorporating the need for the authentication of the receiving node before message decryption. Hence, security against an eavesdropper who tries to listen to the communication between two nodes is promised. Herein, the time of the message generation is chosen as an encryption key. The variation between the messages created at different timings ensures the forward and backward secrecy in the network. A node must know the network architecture, i.e., the number of nodes in the network, the UID of the communicating nodes, and their exact location to locate and decode this hidden key. Hence, an adversary node that is not a part of the network will not locate the key. Also, no additional overhead and exchange mechanism is encountered to communicate the key separately because the key is hidden as a watermark in the message itself. The desired encryption–decryption key lies in the portion of time after the radix point and is taken as a millisecond precision. Such a precision requires a maximum of 10 bits when converted to a binary form. Algorithm 1 shows the pseudo-code for message encryption using LoSWIAE. It makes use of the domino effect to increase the robustness of the scheme against attacks by adversaries. Assuming that the variable length message to be communicated is represented by *var message*, *key time* is a list having an upper bound of 10 bits and stores the generated encryption key. The *msg length* and *key length* store the length of *var message* and *key time*, respectively. For Algorithms 1–6, bit positions in the lists start from index 1 to avoid any confusion.

3.2 Pseudo-watermark Generation

Encrypting the message using time as an encryption key and utilizing the concept of the domino effect (step 8, Algorithm 1) and making the result of the present block dependent on the previous one ensures that the encrypted message depends on temporal credentials as the message itself. The location of the communicating sender and receiver nodes is used for pseudo-watermark generation. This further upgrades the level of security and reduces the cracking probability. Knowing both the time at which the message is generated and the location of nodes is very less likely for an eavesdropper, who is not a part of the network. Furthermore, this algorithm would protect the network from identity replication, masquerade, and Sybil attack for node and message authentication. Algorithm 2 shows the pseudo-code for pseudo-watermark generation using LoSWIAE. For the proposed algorithm, *latitude_s*, *latitude_r*, *longitude_s*, and *longitude_r* store the latitude and longitude at which the sending and receiving nodes (with a precision of up to 10 cm) are placed, respectively, count stores the number of ones inextracted loc, and *pseudo_w* is the desired pseudo-watermark. This requires both the latitude and longitude to be represented in six decimal digits after the radix point. For the addition of floating-point numbers, the value of the radix point is considered to be zero. The maximum number that the latitude and longitude can sum up to is 72 and 74, respectively. A maximum of nine ones can be obtained when converted to binary, including the sum and sign bits.

Algorithm 1. Message encryption

Input: (i) Time from system clock at which the message is generated.
(ii) Original message to be communicated.

Output: Encrypted message having length same as that of the input message.

1. Set *key_time*, *msg_length*, *key_length* = 0.
2. Generate the key for encryption using system clock of the node. Encryption key is the portion of time which conveys th milliseconds.
3. Convert the decimal key obtained in step 2 in binary form and store it in *key_time* for further processing.
4. For *key_time* ≠ null // *calculate the length of key_time*
   Add one to *key_length*
5. For *var_message* ≠ null // *calculate the length of var_message*
   Add one to *msg_length*
6. Starting from the most significant position of the *var_message*, make blocks having block size = *key_length*.
7. If bits left at the rear end of the message are smaller than the size of the block defined in step 6, make a block of those left over bits. In such a case, the size of the last block would be smaller than rest of the blocks.
8. For each block obtained in step 6, perform bit wise XOR operation between the *var_message* bits, *key_time* bits, and the result obtained from the block just prior to the block under consideration. For the first block, consider the previous result to be zero.
9. Replace the original *var_message* with the encrypted message obtained in step 8.
10. Convert the *key_length* into binary.
11. Store the binary representation obtained in step 10 in list time_bits in 4-bit notation. If the number of bits is <4, append 0s before it to make it a 4-bit binary number

Algorithm 2. Pseudo-watermark generation

Input: Latitude and longitude of the communicating nodes from lookup table stored in the node.

Output: w bit pseudo-watermark.

1. Set *latitude_s*, *latitude_r*, latitude_c, *longitude_s*, *longitude_r*, *longitude_c*, *extracted_loc*, *pseudo_w*, *count* = 0
2. Extract latitude of the sender and receiver node from the look up table into the lists *latitude_s* and *latitude_r* respectively.
3. latitude_*c* = *latitude_s* + *latitude_r*
4. For latitude_*c* ≠ null
   n = *n* + 1. //*n stores the length of latitude_c*
5. Initialize lat_*c*=0 // *stores the sum of all the digits of latitude_c*
6. for (*i* = 1, *i* ≤ *n*, *i*++) // *i is the position indicator of latitude_c*
   lat_*c* += latitude_*c*[*i*]
7. Repeat steps 2–6 for longitude of the sender and receiver nodes and store the result in lon_*c*.
8. lat_*c* = lon_*c* + lat_*c*
9. Convert the decimal number obtained in step 8 into binary and store it in the list extracted_loc. Append the sign bit according to sign of the result obtained.
10. Count the number of 1s in the binary number stored in extracted_loc and store it in the variable count.
11. if (count < 3)
    count = 9 - count
12. else
    count = count
13. End if
14. for (*j* = count; extracted_loc[j] ≠ null; *j*--) // *j is the position indicator of extracted_loc.*
    Extract 1 bit from extracted_loc and store it in the list pseudo_w. //*In case number of 1s= =0, then return pseudo_w= 1111.*
    In case len(extracted_loc) < count, append 0's before extracted_loc to make len(extracted_loc)=count.
15. return *pseudo_w*.

3.3 Watermark Generation

The third algorithm manipulates *pseudo_w* according to the UIDs of the sender and receiver nodes, further adding one more dimension to the scheme. Furthermore, this algorithm would ensure that the location of nodes is not compromised if, in the worst case, the message is decrypted and watermark bit positions are traced. For the proposed algorithm, *uid_length* stores UID (bits) size, and *uid sender* and *uid receiver* store the UID number of sender and receiver, respectively. Steps 7–8 of Algorithm 3 start

with the most significant bit (MSB) in the case of $x = 1$. Also, the bit position once XORed cannot be used again for performing the XOR operation. In such a situation, the adjacent element is used in the direction of traversal for completing the assigned task. However, XORed bit has to be counted while traversing the bit positions.

3.4 Watermark Embedding

The watermark generated in Algorithm 3 is embedded in the encrypted message. Algorithm 4 shows the pseudo-code for watermark embedding using LoSWIAE.

Algorithm 3. Watermark generation

Input: (i) w bit pseudo-watermark, (ii) variable length message, and (iii)unique identity numbers of sender node and the *msg_length* (iv)count
Output: w bit watermark
1. *uid_length* = size of (unique identity number of sender node)
2. Set short_msg= int(*Msg_length* / 10)
3. Initialize $x = 1$ // *x is position indicator of pseudo_w*
4. location 1 = (count+ *uid_length*)*pseudo_w* % count
5. location 2 = (count+ *uid_length*)*var_message* % *msg_length*
6. location 3 = (count × *uid_length*)*var_message* % *msg_length*
7. If (short_msg < *uid_length*)
   While ($x \leq$ count) (location 1)th bit *pseudo_w* = (location 2)th bit *var_message* ^ (location1)th bit pseudo_w
   $x = x + 1$
   If (location 1)th bit after XORing = 0
   Subtract the (*uid_length* + count) bits from current location to obtain next locations 1 and 2.
   Else
   Add the (*uid_length* + count) bits to current location to obtain next locations 1 and 2.
8. Else
   (While $x \leq$ count) (location 1)th bit *pseudo_w* = (location 3)th bit *var_message*^ (location1)th bit *pseudo_w*
   $x = x + 1$
   If (location 1)th bit in *pseudo_w* after XORing = 0
   Subtract the (*uid_length* + count) bits from current location to obtain location1 in *pseudo_w* and (*uid_length* × count) bits to obtain location 3 in *var_message*.
   Else
   Add (*uid_length* + *count*) bits to current location to obtain location1 in *pseudo_w* and (*uid_length* × *count*) bits to obtain location3 in *var_message*.
9. Return *pseudo_w*

Algorithm 4. Watermark embedding

Input: w bit pseudo-watermark
(ii) encrypted message
(iii) contents of short_msg and *uid_length*
Output: (*msg_length*)bits watermarked message
1. Initialize $x = 1$ // *x is position indicator of pseudo_w*
2. location 2= (count + *uid_length*) *var_message*% *msg_length*
3. location 3= (count × *uid_length*) *var_message* % *msg_length*
4. If (short_msg < *uid_length*)
   While $x \leq$ count
   Locate the (location 2)th bit in *var_message* //*Start the counting of location 2 from the least significant bit when x = 1*
   Extract the (*x*)th bit from *pseudo_w* //*Start the counting of x from the most significant bit when x = 1.*
   (location 2)th var_message = (location 2)th *var_message* ^(*x*)th *pseudo_w*
   $x = x + 1$
   Subtract (*uid_length+count*) bits from current location to obtain next location2 in *var_message*.
5. Else

While $x \leq count$
Locate the (location 3)th bit in *var_message* //*Start the counting for location3 from the least significant bit when x = 1*
Extract the $(x)^{th}$ bit from *pseudo_w*//*Start the counting of x from the most significant bit when x = 1.*
(location3)th *var_message*= (location3)th *var_message* ^ $(x)^{th}$ *pseudo_w*
$x = x + 1$
Subtract (*uid_length* × count) bits from current location to obtain next location 3 in *var_message.*
6. Return *var_message*

3.5 Key Hint Generation and Embedding

The crux of the proposed scheme lies in Algorithms 5 and 6 for key hint generation and embedding and for key and watermark detection, respectively. Algorithm 5 generates the hint for the receiver node to identify the locations where the decryption key resides. The hint is hidden in the message where it can be obtained only by an authentic node. The specialty of this algorithm is its simple yet robust nature. It makes use of both the geographical and temporal credentials of the node and message, respectively.

3.6 Key and Watermark Detection

Message sent from the sender node is routed according to the receiver UID. The message communicated on the channel is a one-tuple information, i.e., no separation exists between the two portions corresponding to the UID and the message in the transmitted packet on the communicating channel. Nodes that are a part of the network are aware of the number of bits used to represent the UID and can further process the information. Every possible combination corresponding to every possible number of bits in the UID has to be tried because an adversary is unaware of the number of bits required to represent the UID number and the number of bits of the message transmitted. It will require a very high number of trials. For example, if the message transmitted is 32 bits, the combinations of UID bits and message bits could range from 1-bit UID and 31-bit message to 31-bit UID and 1-bit message to anything in between. Algorithm 6 shows the pseudo-code for key and watermark detection.

Algorithm 5. Key hint generation and embedding

Input: UID number of the sender and receiver node (ii) message returned (iii) location of the sender and receiver node (iv) time at which message is generated in binary form (v) *key_length*
Output: Encrypted and watermarked message string.
1. For ($x = 1, x \leq$ uid_length, x++)
Extract the bits in *var_message* obtained after watermark embedding and store them in *uid_s*
2. Extract the UID of the sender and receiver node from the look up table into the lists *uid_sender* and *uid_receiver* respectively.
3. *uid_s =uid_receiver + uid_s* // *Ignore the final carry if the carry is obtained.*
4. Calculate the decimal equivalent of uid_s and store the result in variable deci
5. If (deci × 19 > (*msg_length* + 4+*key_length*))
Deci = int(deci/2)
If (deci × 19 > × (*msg_length* + 4+ *key_length*))
Deci = deci/2
Else
Deci = deci r
6. Set *e* = deci //*e is the position indicator of var_message*
7. uid_sender = uid_sender || time_bits || *key_time*
8. Set *y* = *uid_length*
9. location 5 = *e* + *y*
10. location 6 = *e* + *y* + 1
11. For ($x \leq$ uid_length + 4 + *key_length*)
e = *e* + deci; *x* = *x* + 1

//Extract the x^{th} bit from the uid_sender obtained in step 7 starting from the MSB. Locate (location 5)th bit in the message obtained after the execution of above algorithm.

//Shift bits starting (location 6)th position 1 bit to the right. Append the extracted x^{th} bit at the vacant position created as a result of shifting of bits.

12. return *var_message*

Algorithm 6. Key and watermark detection

Input: Encrypted message from the sender node. Let the encrypted message be represented by enc_msg.

Output: Decryption key, watermarked bits, location of sender and receiver nodes and the original message.

1. Store the first g bits from the *enc_msg* into *uid_s*. Here g=*uid_length* of the receiver node.
2. *uid_s* = *uid_receiver* + *uid_s* // *Ignore the final carry if the carry is obtained.*
3. Calculate the decimal equivalent of *uid_s* and store the result in variable *deci*
4. If (*deci* × 19 > len(*enc_msg*)-*uid_length*)
   Deci = deci/2
   If (*dec* × 19 > × (len(*enc_msg*) - *uid_length*))
   Deci = *deci/*2
   else
   Deci =*deci*
5. Set *e* = *deci*, *y* = *uid_length*
6. location1= e+y-x
7. location 2 = $e + y - x + 1$
8. For ($x = -1$; $x \leq$ uid_length + 2; x++)
   Extract the (location 1)th bit from the *var_message* obtained from the receiver : *uid_sender*.
   e = e + *deci* // Shift bits starting (location 2)th position in *var_message* 1 bit to the left.
   return *e*, *uid_sender*, *var_message*
9. For (z = uid_length + 1; $z \leq$ uid_length + 4; z++) //*z is the position indicator of uid_sender (step 8)*
   Extract bits from uid_sender obtained in step 8 and store it in list *time_bit*.
10. Resize *uid_sender*(step 8) starting from the most significant bit to make it equal to *uid_length*.
11. Convert the binary number so obtained in step9 into decimal and store it in *codeci*.
12. x = *uid_length*+3
13. For ($p = 1$; $p \leq$ *codeci*; p++)
   Extract the (location 1)th bit from the *var_message* obtained in step8 and store it in the list *key_time*.
   Shift bits starting (location 2)th position 1 bit to the left.
   $e = e$ + deci; $x = x + 1$
   // *This is the required decryption key i.e. the time at which the message was generated. Only an authentic node will have an access to location of sender and receiver node.*
14. Store the remaining message in the list *var_message* bit by bit. // *The length var_message would be equal to length (enc_msg)-number of watermarked bits*.
15. Extract the location of receiver node from the location sensors of the node and location of sender node from the look up table present in the node.
16. Repeat steps 3–15 of Fig. 4 and steps 2–9 of Fig. 5 to obtain pseudo_w.
17. Repeat algorithm in Fig. 6 to obtain the positions where XOR operation has been performed with *pseudo_w*
18. Using the property of XOR: If a^$b = c$, then a^$c = b$ and b^$c = a$, obtain the encrypted message.
19. Decrypt the message obtained in step 14 using the decryption key *key_time* obtained in step 13.
20. Original_message = *var_message* ^ *key_time* for each block.

Consider a wireless sensor network with two communicating nodes. A message of 128 bits (randomly chosen to show the efficiency of the network) is to be sent from a node situated at a location (latitude: +78.982765, longitude: +123.808017) to a node situated at a location (latitude: +56.825760, longitude: +87.004679). Let the UID number of sender and receiver nodes be 101011 and 110010, respectively. Let the time at which the message is generated be 16:23:11:451. Examples depicting the use of the proposed scheme stepwise are shown in Figs. 3–7. Let the message to be communicated be: 1010101010101010101010101010101010101010101010101010101010010101010101010101010 10101010101010101010101010101010101010101010.

Following the steps of proposed algorithm, we get
Key_time (451 ms in binary): 111000011
Key_length = 9
Msg_length = 128
The encrypted message can be obtained as explained above. Similarly performing the algorithm for all blocks, following encrypted message is obtained
0100101101111111111011010010000000000100101101111111111011010010000000000
10010110111111111110110100100000000001001011011111111110
Time_bits = 1001

(a)

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Message	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Time	1	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1
Previous result	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0
Encryp. msg.	0	1	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1

(b)

Fig. 3. (a) Message encryption and (b) domino effect on block 2 of the message. Result obtained in block 1 XORed with the contents of block 2.

Following the steps of the proposed algorithm in Fig. 5, we obtain
uid_length = 6
short_msg = 12
count + *uid_length* = 13
count × *uid_length* = 42
Input message from Fig9:
0100101101111111111011010010000000000100101101111111111011010010000000000
10010110111111111101101001000000000010010110111111111110
pseudo_w obtained in Fig. 10: 0 0 1 0 0 1 0
Following step 8 of the proposed algorithm, we get
(0010010) ^ (1101101) = 1111111
Colored bits are obtained by following step 8 of algorithm in Fig. 5 as shown in Fig. 11(a).

(a)

Following the steps of the proposed algorithm in Fig. 4, we obtain *Latitude_s* = +78.982765
Latitude_r = +56.825760
Longitude_s = +123.808 1+ 3+ 5+ 0+ 8+ 0+ 8+ 5+ 2+ 5 =37
Similarly, 017
Longitude_r = +87.004679
Latitude_*c* = 78.982765 + 56.825760 = 135.808525
lat_*c* = Longitude_*c* = 123.808017 + 87.004679 = 210.812696
lon_*c* = 2 + 1 + 0 + 0 + 8 + 1 + 2 + 6 + 9 + 6 = 35
Following steps 8–13 of the proposed algorithm, we obtained
lat_*c* = 35+ 37 = 72
extracted_loc = 01001000
count = 7; pseudo_w: 0010010

(b)

Fig. 4. (a) Watermark generation and (b) pseudo-watermark generation.

Encrypted message obtained is:
0100101101111111111101101001000000000010010110111111111101101001000000000010
0101101111111111101101001000000000010010110111111111110
Location2 = count + *uid_length* = 13
Location3 = count × *uid_length*= 42
Pseudo_w =1111111
Following the steps of proposed algorithm, the following modifications are obtained
0110011011111111111011010010000000000100101111011111111101101001000000000010
0101101111101010011010010000000000100101101111111110

(a)

Message obtained is:
0110001101111111111011010010000000000100101111011111111101101001000000000010
0101101111101010011010010000000000 01001011011111111110
uid_s = 011000 uid_receive r= 110010 uid_s = 001010 deci = 10
Since (10 × 19) > 141, deci = 5 *e* = 5
uid_sender = 1010111001111000011 *y* = 6 for
location 5 can be obtained.
Following the steps of proposed algorithm and appending the bits at locations, we obtain
0110001101111111101111101100100110000100001000100100101011101111111111101101100
1000000000000010010101110111110101001101001000000000010010110111111111110

(b)

Color	Previous Bit after XORing	Bit positions in *var_message*	Bit position in *pseudo_w*	Resultant bit after XORing
	-	42	6	1
	1	84	5	1
	1	126	4	1
	1	40	3	1
	1	82	2	1
	1	124	1	1
	1	38	7	1

(c)

Fig. 5. (a) Watermark embedding, (b) key hint generation and embedding, and (c) calculation of bit positions.

Value of *e*	Value of *y*	Value of *x*	Location 5 in *var_message*	Bit positions in *uid_sender*	Original message and the message sent to the receiver node
5	6	1	11	1	*Original message:* *10101010101010101010101010101010101010101* *01010101010101010101010101010101010101010* *10101010101010101 10101010101010101010* *Encrypted and watermarked message:* *01100011011111110111110110010011000010000100100* *00010101110111111111111011011001000000000000100* *00010111011111101010011010010000000000100101101* *1111111111110* *Message routed along the path toward receiver node:* *11001001100011011111110111110110010011000010000* *10010010001010111011111111111101101100100000000* *00000100010111011111101010011010010000000000100* *10101101111111110*
10	6	2	16	2	
15	6	3	21	3	
20	6	4	26	4	
25	6	5	31	5	
30	6	6	36	6	
–	–	–	–	–	
–	–	–	–	–	
95	6	19	101	19	

Fig. 6. Calculation to obtain bit positions of var_message and uid_sender. Extent of dissimilarity between the original message and the message sent to the receiver node.

Message obtained from the sender node after removing UID of the receiver meant for routing data is given below.
01100011011111110111110110010011000010000100100010101110111111111110110110010000000000000010001011101111110 1001101001000000000010010110111111111110
Following the steps of proposed algorithm in Fig. 3, we obtain
uid_s = 011000; uid_receiver = 110010; uid_s = 001010; deci = 10
Since (10 × 19) > 141, deci = 5, *e* = 5
Location 1 can be obtained as shown in Fig. 15(a).
uid_sender = 1010111001; time_bit = 1001;uid_sender = 101011; $Codec_i$ = 9
Following step 13 of the proposed algorithm, we obtain
Key_time = 111000011
Var_message obtained after removing the watermarked bits is:
01100011011111111110110100100000000001001011110111111111011010010000000000100101101111101010011010010000000000100101101111111110
From the look up table present in the node, we get
Latitude_s = +78.982765; *Latitude_r* = +56.825760; *Longitude_s* = +123.808017;
Longitude_r= +87.004679
Following steps 16 of the proposed algorithm, we get
Pseudo_w= 1111111
uid_length = 6
short_msg = 12
count + *uid_length* = 13
count × *uid_length* = 42
Positions of bits obtained in the message are obtained as shown below.
01100110111111111101101001000000000100101111011111111101101001000000000100101101111110101001101001000000000100101101111111110
Following step 18 of the proposed algorithm, we obtain
01001011011111111111011010010000000001001011011111111101101001000000000100101101111111111011011001000000000100101101111111110
Following steps 19–20 of the proposed algorithm, we obtain
10101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010 which is the original message sans encryption and watermarked bits.

(a)

Value of 'e'	Value of 'y'	Value of 'x'	Location1 in received message
5	6	-1	12
10	6	0	16
15	6	1	20
20	6	2	24
-	-	-	-
-	-	-	-
50	6	8	48

The length of message at different points during the execution of proposed on the original message of length 128 bits will result in the number of bits watermarked (26), length of the message after watermarking (147 bits), and the length of the message communicated (153 bits).

(b)

Fig. 7. (a) Key and watermark detection and (b) calculation of location 1.

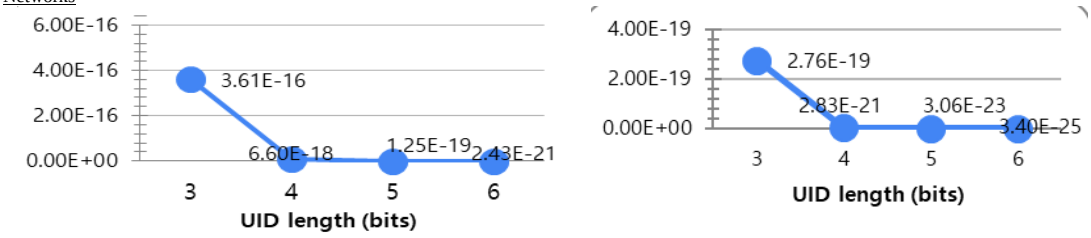
4. Performance Analysis

The proposed LoSWIAE scheme is implemented in Python programming language for random test vectors of 32, 64, and 128 bits. The simulations took a maximum of 1–2 minutes when performed on a machine with an i7 processor and 16 GB RAM. The complexity of the algorithm is roughly 2*n* at both

sender's and receiver's sides based on the location parameters and the UID length. The only information that an adversary can have for an ongoing message from the sender to the receiver node is given by Equation (1) because the number of bits used to represent the UID number of the sender and receiver nodes, location of the nodes, and the time at which the message is generated is scalable. Thus, the total length is a function of (*uid length*, count, *key time*, and *msg length*). Consequently, only the total number of bits transmitted is available to an adversary eavesdropping on the communicated message, while the number of possible combinations of individual terms sums up to the left-hand side of the Equation (1) along with the correct permutation of bits is very high. For instance, all the combinations (4, 4, 6, and 135), (8, 3, 8, and 126), and (12, 9, 9, and 111) sum up to a total of 153 bits (total length for illustration in Section 4).

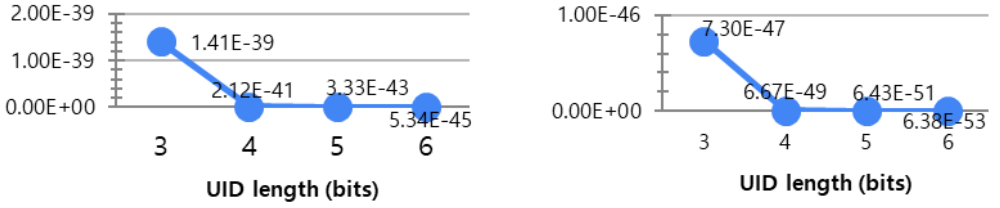
LoSWIAE performance is analyzed in terms of cracking probability and communication overhead. Consider a case in which information is being communicated from an isolated military base A to a location headquartered at B at a far-off distance. For an adversary to decrypt, manipulate, and send the message as an authentic node located in base camp A, the network architecture, the number of nodes in the network, the location of communicating nodes, the time at which the message was generated, and the UID numbers of the communicating nodes will have to be known. Guessing this high precision (centimeters) in an area of a few kilometers (assuming A to be spread over a few kilometers) is highly improbable. As explained earlier, using the time stamp as the key makes it difficult to deduce the encryption–decryption key and decipher the message. The number of trials required would be $2m$ where m is the length of the message communicated even in the case of a brute force attack. Furthermore, if an eavesdropper gets hold of the message and tries to make even subtle changes in the message, the final result would be drastically modified due to the domino nature of the algorithm, thus ensuring an integrity check. The discovery of the exact location of the sender node is a challenging task and LoSWIAE would prove to be a paragon in most of the applications of WSN with geographical variations like underground coal mines, hazardous gas stations, oceanographic monitoring, and so on.

Cracking probability: Cracking probability refers to the probability that an adversary may be able to decrypt the communicated message and manipulate it to originate from an authentic node. The cracking probability of LoSWIAE is computed as follows: the number of bits (n) required for UID of the receiver can be calculated by Equation (2), and the probability that an adversary would correctly obtain the number of bits required to represent the UID of the receiver node can be calculated by Equation (3). For intra-network communication, the number of bits required to represent the UID of the sender node can be calculated. For an m bit message communicated, the probability of obtaining the correct positions of the sender's UID for the correct number of UID receiver's bits can be calculated by Equation (4). The probability of obtaining j correct positions for the number of time bits can be calculated by Equation (5). Moreover, the probability of obtaining k correct positions of the watermarked key time bits can be calculated by Equation (6). Furthermore, the probability of obtaining one correct position of the bits in which the XOR operation is performed can be calculated by Equation (7). Hence, from Equations (3)–(7), the cracking probability is obtained as calculated by Equation (8) where u is the number of bits in the UID of the receiver node. The cracking probability for the given illustration of the 128-bit message and 6-bit UID, considering the location and time at which the message is generated, is found to be 7×10^{-55}. The maximum, minimum, and average cracking probabilities for 32- and 64-bit combinations of inputs are shown in Figs. 8–10. A similar trend of cracking probabilities is observed for the original message length of 128- and 256-bit as shown in Fig. 11.



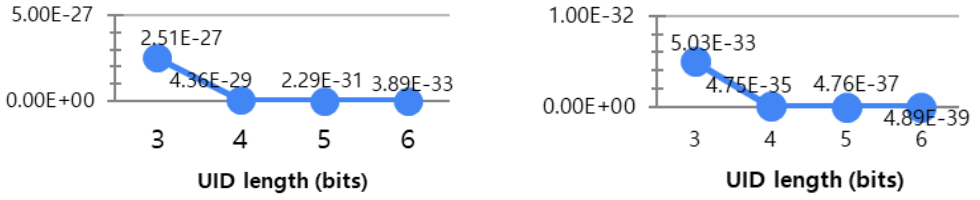
(a) (b)

Fig. 8. Maximum cracking probability vs. UID length for (a) 32-bit message and (b) 64-bit message.



(a) (b)

Fig. 9. Minimum cracking probability vs. UID length for (a) 32-bits message and (b) 64-bits message.



(a) (b)

Fig. 10. Average cracking probability vs. UID length for (a) 32-bits message and (b) 64-bits message.

Total number of bits transmitted =

(2 × number of bits representing UID number of sender and receiver nodes + number of bits representing location nodes + number of bits representing the time at which the message is generated + 4 + length of original message) (1)

$$\text{Node} \geq \text{ceiling}\left(\frac{log\ (number\ of\ nodes\ in\ the\ network)}{log\ 2}\right) \quad (2)$$

$$1/n \quad (3)$$

$$\sum_{i=1}^{n} 1/(m-u)p_i \quad (4)$$

$$\sum_{k=1}^{4} 1/(m-u-1)p_k \quad (5)$$

$$\sum_{k=3}^{10} 1/\,(m-u-i-4-k)p_k \quad (6)$$

$$\sum_{l=3}^{count} 1/(m-u-i-4-k)p_l \quad (7)$$

$$((1/\mathrm{n})\sum_{i=1}^{n} 1/(m-u)p_i\,[\sum_{j=1}^{4} 1/(m-u-i)p_j + \left[\sum_{k=1}^{10} 1/\,(m-u-i)p_k * \sum_{l=3}^{count} 1/(m-u-i-4-k)p_l\right]]100) \quad (8)$$

(a)

msg. (bits)	UID lgth (bits)	Max. crack. prob.	Min. crack. prob.	Min. Tx Over. (uJ)	Max. Tx Over (uJ)	Min.Rx Over. (uJ)	Max. Rx over. (uJ)	Avg. crack. Prob.	Avg. Tx Over. (uJ)	Avg. Rx. Over. (uJ)
128	3	1.71E-22	2.30E-54	83.4	88.8	93.13	99.16	2.01E-38	86.1	96.14
	4	9.38E-25	1.18E-56	84.6	90	94.47	100.5	5.29E-40	87.3	97.48
	5	5.44E-27	6.45E-59	85.8	91.2	95.81	101.8	5.95E-43	88.5	98.825
	6	3.26E-29	3.63E-61	87	92.4	97.15	103.2	3.45E-45	89.7	100.17
256	3	9.48E-26	5.20E-62	160.2	165.6	178.89	184.9	7.34E-44	162.9	181.9
	4	2.68E-28	1.42E-64	161.4	166.8	180.23	186.2	2.05E-46	164.1	183.2
	5	8.07E-31	4.14E-67	162.6	168	181.57	187.6	6.11E-49	165.3	184.58
	6	2.51E-33	1.25E-69	163.8	169.2	182.91	188.9	1.88E-51	166.5	185.93

(b)

Fig. 11. (a) Cracking probability and communication overheads. (b) Transmission (Tx) and reception (Rx) for various combination of inputs.

The cracking probability, as observed, varies from a maximum to a minimum for a similar combination of a number of bits representing the message and UID number of the node. This is due to the variation in the number of watermarked bits. The number of bits watermarked depends on the UID number of the sender and receiver node, the location of the node, and the time at which the message is generated. The cracking probability for various combinations of lengths is given in Table 1 and calculated using Equation (8) as derived.

Table 1. Comparison of cracking probability and overheads of the proposed with existing schemes

Scheme	Msg length (bits)	UID length (bits)	Crack prob	Tx over (μJ)	Rx over (μJ)
LoWaNA [28]	128	6	4.00E-13	84.40	93.80
	64	5	3.41E-09	44.4	49.5
	32	4	7.24E-06	24	26.8
LoSWIAE	128	6	3.45E-45	89.7	100.16
	64	5	4.76E-37	50.1	55.94
	32	4	4.36E-29	29.7	33.16

Communication overhead of LoSWIAE refers to the number of bits transmitted from the sender node to the receiver node. The overhead for LoSWIAE includes bits required to transmit the vital information, the watermarked bits, the decryption key, and the header (includes the UID of the receiver node according to the network model of the proposed scheme). As per MICAz specifications, to transmit and receive 1 bit of information, 0.60 and 0.67 μJ of energy is consumed, respectively [21]. Communication overhead for various combinations of inputs is shown in Fig. 12. For illustration, the transmission and reception overhead are 91.8 and 102.51 μJ, respectively.

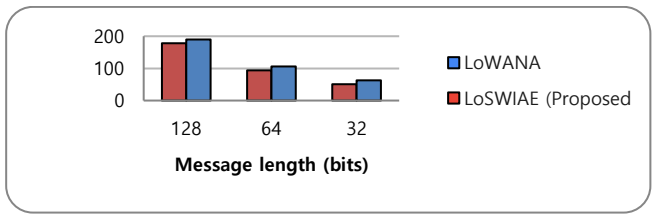


Fig. 12. Comparison of communication overhead of LoSWIAE with LoWaNA [28].

5. Comparative Analysis

Due to the dearth of studies in the field of authentication and encryption in MWSN using watermarking and the lack of comparable data, LoSWIAE is compared with a recently published scheme LoWaNA [28]. Fig. 11 shows the comparative evaluation of LoSWIAE in terms of cracking probability and communication overhead. The proposed scheme is found to be more efficient by a factor of 1,032 for a message length of 128 bits, 1,028 for a message length of 64 bits, and 1,023 for a message length of 32 bits compared to LoWaNA [25]. This high efficiency is achievable in a trade-off with an infinitesimal increase in communication overhead (Fig. 9). In terms of the security analysis: the use of the *key_time* as the time information of message generation helps protect against the Replay attack. The sensor node impersonation and DDoS attacks are addressed by the improved cracking probability. The MITM attack is addressed by the XORing function for identity checks explained earlier.

6. Conclusion

This paper proposes a scalable and low overhead integrated authentication and encryption scheme based on watermarking. The proposed scheme amalgamates the advantages of authentication and encryption through watermarking and is scalable, reversible, and self-sustainable which sets it apart from other authentication and encryption schemes in MWSNs. Algorithms of LoSWIAE proposed in the scheme make use of single-clock cycle operations (e.g., XOR, shift, insertion, and extraction), thus making the scheme computationally inexpensive. The proposed scheme is devised with an accuracy of a millisecond of temporal credentials and a resolution of a few centimeters in geographical credentials. The performance of the scheme is analyzed in terms of cracking probability and communication overhead. This opens an insight into lightweight and scalable watermarking-based algorithms for WSN that provides security features proven by computationally expensive cryptographic algorithms. However, owing to the vulnerability of the physical nodes, any adversary using artificial intelligence can steal their sensitive information which may pose further challenges. Therefore, future work may include testing the proposed approach for PUF-enabled IoMT applications.

Acknowledgements

Authors acknowledges to the University Kebangsaan Malaysia for supporting this work.

Author’s Contributions

Conceptualization, HV, MKH, HS, SI, MSM. Funding acquisition, HV, MKH, HS, SI, MSM. Investigation and methodology, HV, AKMAH, MKH, MSA, HA. Supervision, HV, MKH, HS, SI, MSM. Writing of the original draft, HV, MKH, HS, SI, MSM. Writing of the review and editing, HV, MKH, HS, SI, MSM. Software, HV, MKH, HS, SI, MSM. Data curation, AKMAH, MSA, HA.

Funding

This research was supported by the GP-2021-K023208 grant through the National University of Malaysia (UKM), Also, this research was supported by Taif University Researchers Supporting Project number (TURSP-2020/216), Taif University, Taif, Saudi Arabia.

Competing Interests

The authors declare that they have no competing interests.

References

[1] R. Hassan, F. Qamar, M. K. Hasan, A. H. M. Aman, and A. S. Ahmed, "Internet of Things and its applications: a comprehensive survey," *Symmetry*, vol. 12, no. 10, article no. 1674, 2020. https://doi.org/10.3390/sym12101674

[2] E. S. Ali, M. K. Hasan, R. Hassan, R. A. Saeed, M. B. Hassan, S. Islam, N. S. Nafi, and S. Bevinakoppa, "Machine learning technologies for secure vehicular communication in internet of vehicles: recent advances and applications," *Security and Communication Networks*, vol. 2021, article no. 8868355, 2021. https://doi.org/10.1155/2021/8868355

[3] I. Memon, M. K. Hasan, R. A. Shaikh, J. Nebhen, K. A. A. Bakar, E. Hossain, and M. H. Tunio, "Energy-efficient fuzzy management system for internet of things connected vehicular ad hoc networks," *Electronics*, vol. 10, no. 9, article no. 1068, 2021.

[4] S. C. Mukhopadhyay and N. K. Suryadevara, "Internet of Things: challenges and opportunities," *Internet of Things*, 2014. https://doi.org/10.1007/978-3-319-04223-7_1

[5] J. H. Park, S. Rathore, S. K. Singh, M. M. Salim, A. E. Azzaoui, T. W. Kim, Y. Pan, and J. H. Park, "A comprehensive survey on core technologies and services for 5G security: taxonomies, issues, and solutions," *Human-centric Computing and Information Sciences*, vol. 11, article no. 3, 2021. https://doi.org/10.22967/HCIS.2021.11.003

[6] S. Singh, P. K. Sharma, S. Y. Moon, D. Moon, and J. H. Park, "A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 4543-4574, 2019.

[7] Q. Dang, "Changes in federal information processing standard (FIPS) 180-4, secure hash standard," *Cryptologia*, vol. 37, no. 1, pp. 69-73, 2013.

[8] V. Manjula and C. Chellappan, "The replication attack in wireless sensor networks: analysis and defenses," in *Advances in Network and Communications*. Heidelberg, Germany: Springer, 2021, pp. 169-178.

[9] R. Perlman and C. Kaufman, "Byzantine robustness, hierarchy, and scalability," in *Proceedings of 2013 IEEE Conference on Communications and Network Security (CNS)*, National Harbor, MD, 2013, pp. 242-250.

[10] M. K. Hasan, M. Shafiq, S. Islam, B. Pandey, Y. A. Baker El-Ebiary, N. S. Nafi, R. Ciro Rodriquez, and D. E Vargas, "Lightweight cryptographic algorithms for guessing attack protection in complex Internet of Things applications," *Complexity*, vol. 2021, article no. 5540296, 2021. https://doi.org/10.1155/2021/5540296

[11] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 2018.

[12] S. Medileh, A. Laouid, R. Euler, A. Bounceur, M. Hammoudeh, M. AlShaikh, A. Eleyan, and O. A. Khashan, "A flexible encryption technique for the Internet of Things environment," *Ad Hoc Networks*, vol. 106, article no. 102240, 2020. https://doi.org/10.1016/j.adhoc.2020.102240

[13] G. Xu, W. Shen, and X. Wang, "Applications of wireless sensor networks in marine environment monitoring: a survey," *Sensors*, vol. 14, no. 9, pp. 16932-16954, 2014.

[14] V. Bapat, P. Kale, V. Shinde, N. Deshpande, and A. Shaligram, "WSN application for crop protection to divert animal intrusions in the agricultural land," *Computers and Electronics in Agriculture*, vol. 133, pp. 88-96, 2017.

[15] M. Winkler, M. Street, K. D. Tuchs, and K. Wrona, "Wireless sensor networks for military purposes," *Autonomous Sensor Networks*, pp. 365-394, 2012. https://doi.org/10.1007/5346_2012_40

[16] A. H. Moon, U. Iqbal, and G. M. Bhat, "Light weight authentication framework for WSN," in *Proceedings of 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, India, 2016, pp. 3099-3105.

[17] G. Zhang, L. Kou, L. Zhang, C. Liu, Q. Da, and J. Sun, "A new digital watermarking method for data integrity protection in the perception layer of IoT," *Security and Communication Networks*, vol. 2017, article no. 3126010, 2017. https://doi.org/10.1155/2017/3126010

[18] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, 1987.

[19] A. Kamili, N. N. Hurrah, S. A. Parah, G. M. Bhat, and K. Muhammad, "DWFCAT: dual watermarking framework for industrial image authentication and tamper localization," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5108-5117, 2021.

[20] W. Wang, H. Xu, M. Alazab, T. R. Gadekallu, Z. Han, and C. Su, "Blockchain-based reliable and efficient certificateless signature for IIoT devices," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 7059-7067, 2022.

[21] H. Xiong, C. Jin, M. Alazab, K. H. Yeh, H. Wang, T. R. Gadekallu, W. Wang, and C. Su, "On the design of blockchain-based ECDSA with fault-tolerant batch verification protocol for blockchain-enabled IoMT," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 1977-1986, 2022.

[22] J. C. S. Sicato, S. K. Singh, S. Rathore, and J. H. Park, "A comprehensive analyses of intrusion detection system for IoT environment," *Journal of Information Processing Systems*, vol. 16, no. 4, pp. 975-990, 2020.

[23] R. M. Swarna Priya, P. K. R. Maddikunta, M. Parimala, S. Koppu, T. R. Gadekallu, C. L. Chowdhary, and M. Alazab, "An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture," *Computer Communications*, vol. 160, pp. 139-149, 2020.

[24] M. H. Mahalat, D. Karmakar, A. Mondal, and B. Sen, "PUF based secure and lightweight authentication and key-sharing scheme for wireless sensor network," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 18, no. 1, article no. 9, 2021. https://doi.org/10.1145/3466682

[25] W. Wang, Q. Chen, Z. Yin, G. Srivastava, T. R. Gadekallu, F. Alsolami, and C. Su, "Blockchain and PUF-based lightweight authentication protocol for wireless medical sensor networks," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8883-8891, 2022.

[26] National Institute of Standards and Technology, "Federal Information Processing Standards Publications (FIPS PUBS) Index," 1997 [Online]. Available: https://doi.org/10.6028/nist.fips.58-1997.

[27] National Institute of Standards and Technology, "NIST centennial sessions," 2001 [Online]. Available: https://doi.org/10.6028/nist.ir.6769.

[28] A. Sen, T. Chatterjee, and B. DasBit, "LoWaNA: low overhead watermark based node authentication in WSN," *Wireless Networks*, vol. 22, no. 7, pp. 2453-2467, 2016.

[29] A. Ghazvini, S. N. H. S. Abdullah, M. K. Hasan, and D. Z. A. B. Kasim, "Crime spatiotemporal prediction with fused objective function in time delay neural network," *IEEE Access*, vol. 8, pp. 115167-115183, 2020.

[30] M. K. Hasan, S. Kamil, M. Shafiq, S. Yuvaraj, E. S. Kumar, R. Vincent, and N. S. Nafi, "An improved watermarking algorithm for robustness and imperceptibility of data protection in the perception layer of Internet of Things," *Pattern Recognition Letters*, vol. 152, pp. 283-294, 2021.