JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 10, ISSUE 03, 2023

Integrity Verification and Data Security in Wireless Sensor Networks

B. Srinivas Reddy[1], Bhavana Kolli[2], Manish Manala[2], Sampeta Jaya Naga Aravind[2],

S Bharath Reddy[2]

[1,2]Department of Information Technology

[1,2] CMR Engineering College, Kandlakoya, Medchal, Hyderabad.

ABSTARCT

In recent years, wireless sensor networks (WSNs) have become a useful tool for environmental monitoring and information collection due to their strong sensory ability. Whereas WSNs utilize wireless communication and is usually deployed in an outdoors environment, which make them vulnerable to be attacked and then lead to the privacy disclosure of the monitored environment. SUM, as one common query among the queries of WSNs, is important to acquire a high-level understanding of the monitored environment and establish the basis for other advanced queries. In addition, now-a-days WSNs are using everywhere such as road traffic monitoring, CCTV home monitoring and many more. They are small and tiny devices which sense the data from its environment and report to the centralized server for remote monitoring using Internet of Things (IoT) network connections. But sometimes malicious attackers can intrude or intercept network connection to alter messages and these altered messages will report to centralized server which many take wrong decision based on received data.

Therefore, to overcome from such issues many encryption technologies were introduced which are based public or private keys and if these keys exposed then the data will be exposed to attacker. To tackle such problem, this project implementing the data integrity verification using Chaotic technique and data security using homomorphic encryption. In proposed technique, the system will send a verification hash code along with encrypted message to the receiver. Next, receiver will decrypt the received message and then it will re-generate the verification hash code. Finally, the verification will be successful if this received, and the generated hash code matches.

Keywords: Integrity verification, Data security, Wireless sensor network.

1. INTRODUCTION

In most of the real-life applications such as battle-field surveillance and forest fire detection using wireless sensor networks (WSNs), a group of sensor nodes are densely and randomly deployed and are left unattended under hazardous conditions. The sensor nodes collaboratively monitor events such as movement of enemy troops in battle-field and communicate with each other over the wireless channel [1]. The sensor nodes can directly send the observations to a central trusted powerful entity, called the base station, or through the intermediate nodes such as cluster heads. The wireless nature of the communication channel and the unattended deployment in difficult terrains leave the network vulnerable to various attacks including node capture. On the other hand, the inherent resource constraints of sensor nodes restrict using expensive security solutions for WSN [2]. Although researchers have addressed various aspects of WSN security such as secure key management, secure localization, and data aggregation, the problem of node capture attack is still a major concern in WSN. In a node capture attack [3], an attacker gets hold of a node physically and then reprograms and redeploys the node. The severity of the attack depends on the time and resources available with the attacker. Various measures for increased resilience to node capture attack have been proposed in the literature through key management schemes; however, detection of node capture attack remains a challenging [4] research problem. As the nodes in a WSN continuously interact with their neighbouring nodes, an unusual absence resulting from physical capture of a node can be noticed with periodic monitoring. Many protocols have been proposed for detection of node capture by continuous monitoring. However, with the malicious neighbor collusion, the no captured malicious nodes may intentionally report an absent node to be present in the network [5].

2. LITERATURE SURVEY

Desai et. al proposes a trust evaluation methodology that evaluates a node-level trust by using an internal resource of a node. It is a completely mediating technique which is independent of network topology and second-hand information. The challenge response method enables a node to evaluate trust for itself; and with its peer-node/s with which it intends to interact using the proposed self-scrutiny and self-attestation algorithms, respectively. The efficacy of the proposed software-based methodology and the algorithms is demonstrated with the actual implementation on sensor nodes. The ability to counter attack-scenarios along with the analysis exhibits the merit of the proposed work. The average values of the observed results illustrate the consistency and robust performance of the proposed trust evaluation algorithms.

Miranda et. al proposed a software-defined security framework that combines intrusion prevention in conjunction with a collaborative anomaly detection systems. Initially, an IPS-based authentication process is designed to provide a lightweight intrusion prevention scheme in the data plane. Subsequently, a collaborative anomaly detection system is leveraged with the aim of supplying a cost-effective intrusion detection solution near the data plane. Moreover, to correlate the true positive alerts raised by the sensor nodes in the network edge, a Smart Monitoring System (SMS) is exploited in the control plane. The performance of the proposed model is evaluated under different security scenarios as well as compared with other methods, where the model's high security and reduction of false alarms are demonstrated.

Cui et. al proposed a secure energy-saving data aggregation scheme designed for the large-scale WSNs. We employ Okamoto-Uchiyama homomorphic encryption algorithm to protect end-to-end data confidentiality, use MAC to achieve in-network false data filtering, and utilize the homomorphic MAC algorithm to achieve end-to-end data integrity. Two popular IEEE 802.15.4-compliant wireless sensor network platforms, Tmote Sky and iMote 2 have been used to evaluate the efficiency and feasibility of our scheme. The results demonstrate that our scheme achieved better performance in reducing energy consumption. Moreover, system delay, especially decryption delay at the base station, has been reduced when compared to other state-of-art methods.

Agarkar et. al presents a post-quantum security solution using lattice cryptography. The proposed solution is applying Learning with Errors over Rings (R-LWE) for encryption of data in the data aggregation process at the gateway module of WSN. Security analysis shows that the proposed scheme provides confidentiality, integrity and authenticity during communication. Performance analysis shows that the proposed scheme is lightweight and shows better performance compared to Elliptic Curve Elgamal (ECEG) cryptography-based scheme and symmetric homomorphic based scheme.

Liu et. al presents a comprehensive review of secure DA (SDA) in WSNs, including its security goals together with the existing problems. The traditional network topologies as well as new emerging ones are discussed and compared in order to indicate the application scenes and security levels of different topologies. Meanwhile, the contrastive analyses of security strategies are presented which divides SDA protocols into five categories according to different security mechanisms, security goals, and network topologies.

Raja Waseem Anwar et. al proposed an efficient Belief based trust evaluation mechanism (BTEM) which isolates the malicious node from trust-worthy nodes and defend against Bad-mouth, On–Off and Denial of Service (DoS) attacks. Bayesian estimation approach is used in gathering direct and In-direct trust values of the sensor nodes which further considers the correlation of the data collected over the time and then estimate imprecise knowledge in decision making for secure delivery of data thus avoiding the malicious nodes. Compared with existing approaches, the proposed BTEM performs better in the detection of malicious node (MN), with lesser delay and improved network throughput.

Fang et. al proposed a Gaussian distribution-based comprehensive trust management system (GDTMS) for F-IWSN. Furthermore, in its trust decision, the grey decision making is introduced to achieve the trade-off between security, transmission performance and energy consumption. The proposed trade-off can effectively select the secure and robust relay node, namely, a trust management-based secure routing scheme. In addition, the proposed schemes are also applicable to defending against bad mouthing attacks. Simulation results show that, the comprehensive performance of GDTMS is better than other similar algorithms. It can effectively prevent the appearance of network holes, and balance the network load, promote the survivability of the network.

Lyu et. al proposed a selective authentication-based geographic opportunistic routing (SelGOR) to defend against the DoS attacks, meeting the requirements of authenticity and reliability in WSNs. By analyzing statistic state information (SSI) of wireless links, SelGOR leverages an SSI-based trust model to improve the efficiency of data delivery. Unlike previous opportunistic routing protocols, SelGOR ensures data integrity by developing an entropy-based selective authentication algorithm, and is able to isolate DoS attackers and reduce the computational cost. Specifically, we design a distributed cooperative verification scheme to accelerate the isolation of attackers. This scheme also makes SelGOR avoid duplicate data transmission and redundant signature verification resulting from opportunistic routing. The extensive simulations show that SelGOR provides reliable and authentic data delivery, while it only consumes 50% of the computational cost compared to other related solutions.

3. PROPOSED SYSTEM

3.1 Chaotic Technique

This project implementing the data integrity verification using Chaotic technique and data security using homomorphic encryption. In proposed technique, the system will send a verification hash code along with encrypted message to the receiver. Next, receiver will decrypt the received message and then it will re-generate the verification hash code. Finally, the verification will be successful if this received, and the generated hash code matches.

Chaos-based encryption algorithms offer many advantages over conventional cryptographic algorithms, such as speed, high security, affordable overheads for computation, and procedure power.

Chaotic techniques are a class of mathematical algorithms used for cryptography and data encryption. They are based on chaotic systems, which are deterministic systems that exhibit seemingly random and unpredictable behavior. Chaotic techniques use the inherent randomness of chaotic systems to generate cryptographic keys or encrypt data.

The basic idea behind chaotic techniques is to use a chaotic system to generate a sequence of numbers that can be used as a cryptographic key or to encrypt data. The sequence of numbers generated by a chaotic system can be highly unpredictable, which makes it difficult for an attacker to guess the key or decrypt the data.

Chaotic techniques have several advantages over traditional cryptographic techniques. They are more resistant to attacks based on mathematical algorithms, and they can be more efficient and faster than traditional techniques. However, they also have some limitations, such as sensitivity to initial conditions, which can make them difficult to implement in practice.

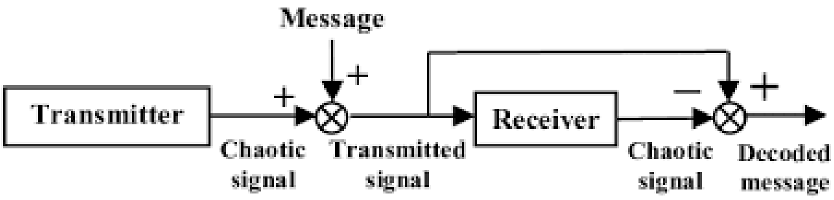


Fig. 1: Chaotic technique.

Some examples of chaotic techniques include chaotic maps, chaotic oscillators, and chaotic neural networks. These techniques have been used in various applications, such as secure communication, image encryption, and digital watermarking. While chaotic techniques are not yet widely used in practice, they have the potential to be an important tool for secure data encryption and communication in the future.

3.2 Homomorphic encryption

Homomorphic encryption is a type of encryption that allows computation to be performed on encrypted data without first decrypting it. In other words, it allows operations to be performed on encrypted data while keeping the data encrypted throughout the computation process. The result of the computation is also encrypted, and can only be decrypted by the authorized user with the appropriate decryption key.

Homomorphic encryption is useful in situations where data privacy and security are critical concerns. It allows sensitive data to be stored and processed securely, without the need for decryption and re-encryption at each step of the computation process.

There are two main types of homomorphic encryption: fully homomorphic encryption (FHE) and partially homomorphic encryption (PHE). FHE allows for any computation to be performed on encrypted data, while PHE allows for only a limited set of computations to be performed on encrypted data.

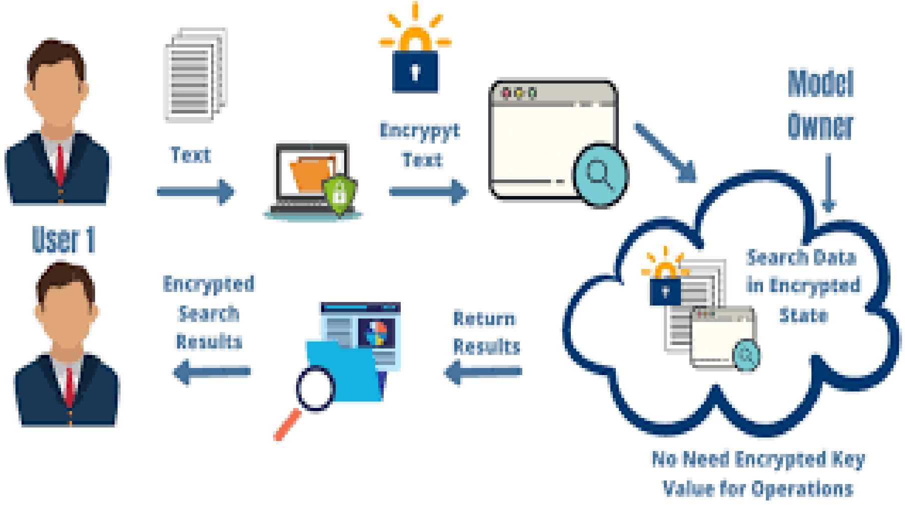

User 1
Text
Encrypyt Text
Model Owner
Search Data in Encrypted State
No Need Encrypted Key Value for Operations
Return Results
Encrypted Search Results

Fig. 2: Hormonic encryption.

While homomorphic encryption has many potential benefits, it also has some limitations. It can be computationally expensive and slow, which can make it impractical for certain applications. It also requires a high level of expertise and knowledge to implement and use correctly.

Despite its limitations, homomorphic encryption has the potential to revolutionize the field of data privacy and security by allowing secure computation to be performed on sensitive data without compromising its confidentiality. It is a promising area of research and development, and has the potential to be an important tool for securing sensitive data in the future.

4. RESULTS AND DISCUSSION

Now-a-days wireless sensor networks are using everywhere for monitoring such as road traffic monitoring, CCTV home monitoring and many more. WSN are small, tiny devices which sense data from its environment and report to centralized server for monitoring using IOT network connections. Sometime some malicious attackers can intrude or intercept network connection to alter messages and this altered message will report to centralized server which many take wrong decision based on received data. To overcome from such issues many encryption technologies were introduced which are based public or private keys and if these keys exposed then data will be exposed to attacker. To tackle such problem, we are adding data Integrity Verification using Chaotic technique and data security using Homomorphic encryption.

In proposed technique we are sending Verification Hashcode along with encrypted message and receiver will receive message and then decrypt the message and then re-generate Verification Hashcode and if received and generated Hashcode matched then verification will be successful.

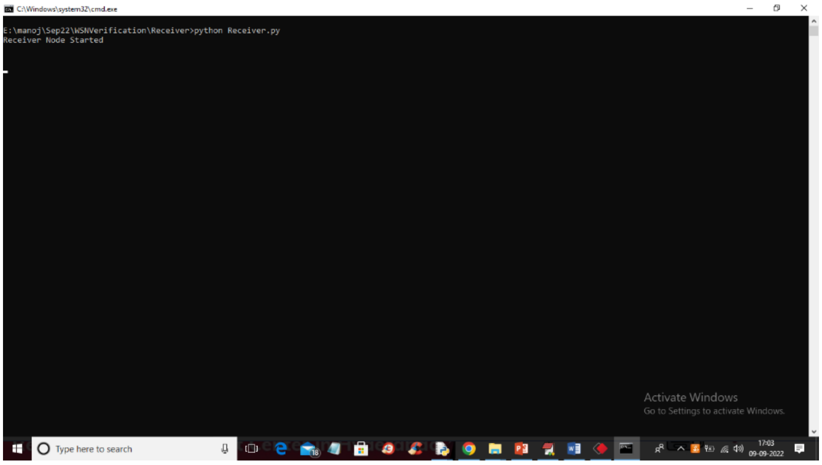
To generate Verification code, we are taking sensor MESSAGE & Secret Key as Input and then convert and pad message to BINARY and then split binary data into blocks and then convert those blocks to 512 randomized hash code and in below screen we are showing code for Chaotic Hashcode Integrity Algorithm.

To implement this project, we have designed following modules

1) Sender: sender will select source and destination from simulation and then generate secret key and then calculate integrity code and then encrypt message and send to receiver
2) Receiver: receiver will receive message with Integrity code and then decrypt message and then regenerate Integrity code from decrypted message and if received and re-generated signature or integrity code matches then verification is successful

Screen Shots

To run project first double, click on 'run.bat' file from 'Receiver' folder to get below screen

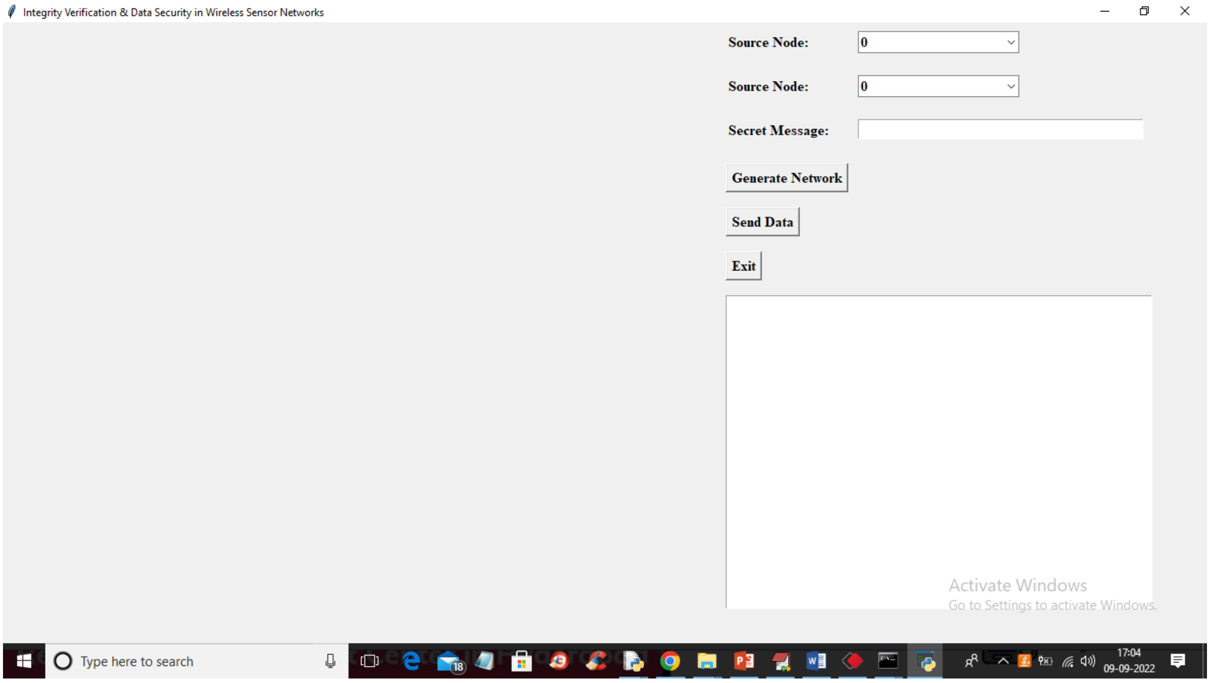


C:\Windows\system32\cmd.exe

```
E:\manoj\Sep22\WSNVerification\Receiver>python Receiver.py
Receiver Node Started
```

Activate Windows
Go to Settings to activate Windows.
Type here to search
17:03
09-09-2022

In above screen 'Receiver' application started and now let the application running and then double click on 'run.bat' file from 'Sender' folder to get below screen

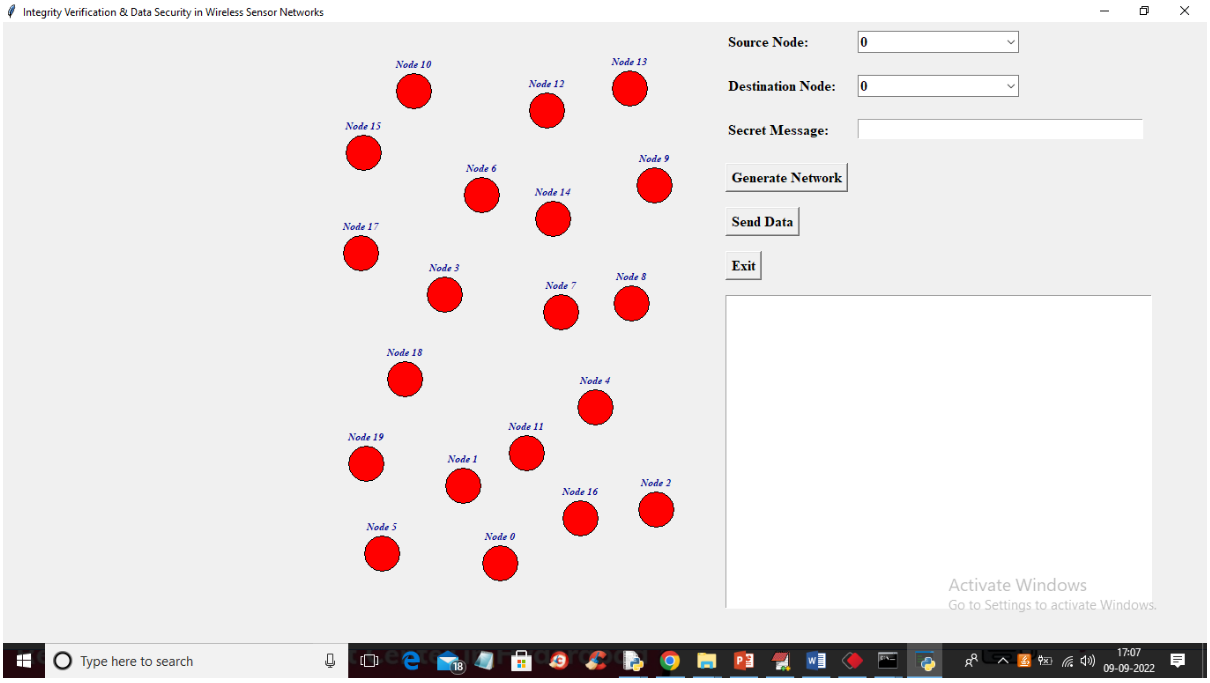

Integrity Verification & Data Security in Wireless Sensor Networks

Source Node: 0

Source Node: 0

Secret Message:

In above screen first click on 'Generate Network' button to create WSN network and get below output

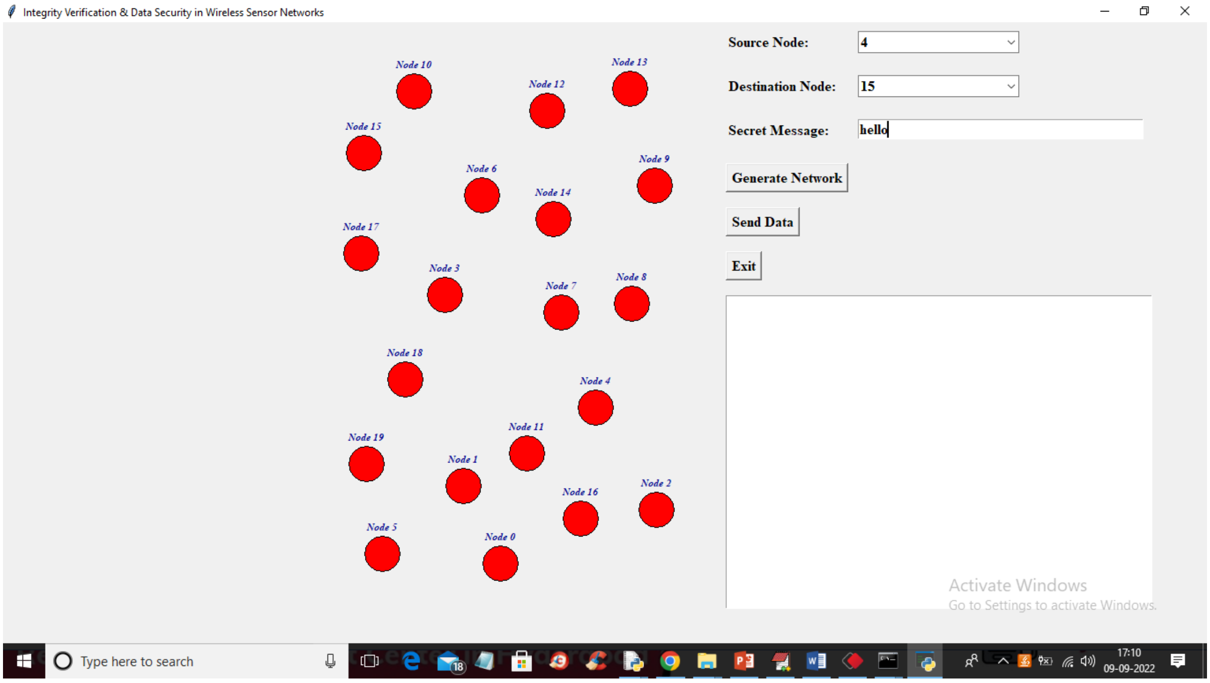

Integrity Verification & Data Security in Wireless Sensor Networks

Source Node: 0

Destination Node: 0

Secret Message:

In above screen all red colour circles are the WSN nodes and now select any source and destination from source and destination drop down box and then enter some message in text field and then press 'Send Data' button to encrypt and send data to destination



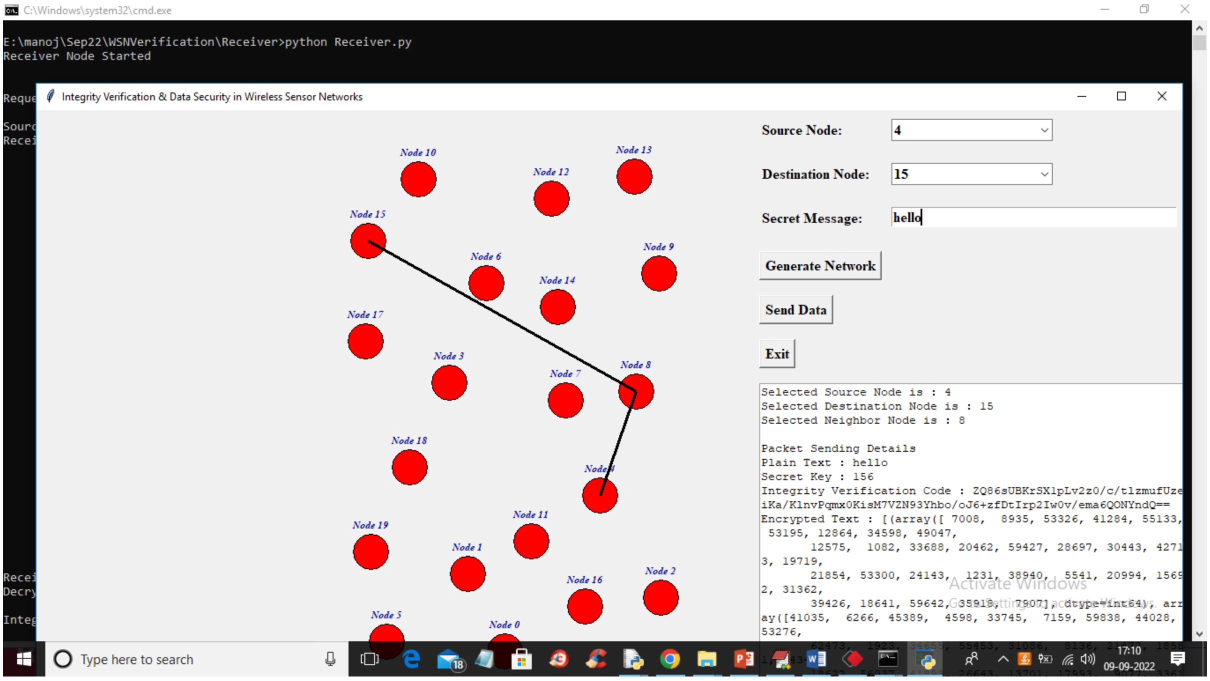
Integrity Verification & Data Security in Wireless Sensor Networks

Source Node: 4

Destination Node: 15

Secret Message: hello

In above screen I selected source node as 4 and destination node as 15 and then entered message as ‘hello’ and then press ‘Send Data’ button to get below output



C:\Windows\system32\cmd.exe

```
E:\manoj\Sep22\WSNVerification\Receiver>python Receiver.py
Receiver Node Started

Reque

Sourc
Recei

Recei
Decry

Integ
```

Integrity Verification & Data Security in Wireless Sensor Networks

Source Node: 4

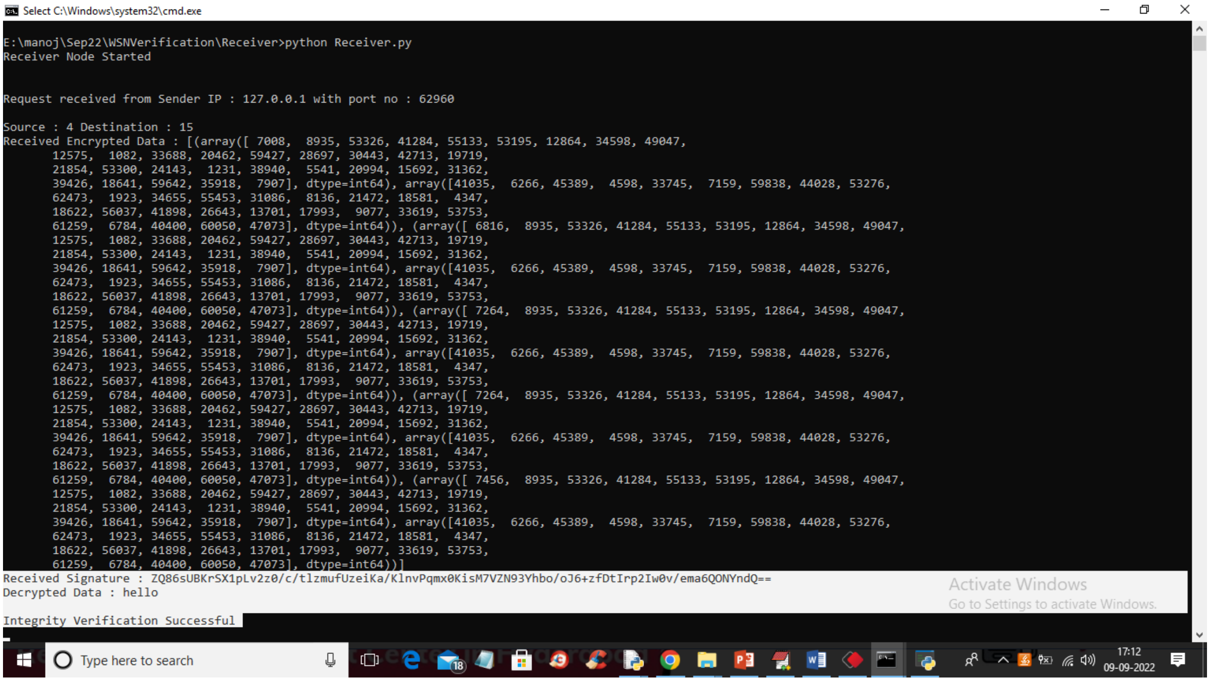
Destination Node: 15

Secret Message: hello

```
Selected Source Node is : 4
Selected Destination Node is : 15
Selected Neighbor Node is : 8

Packet Sending Details
Plain Text : hello
Secret Key : 156
Integrity Verification Code : ZQ86sUBKrSXlpLv2z0/c/tlzmufUze
iKa/KlnvPqmx0KisM7VZN93Yhbo/oJ6+zfDtIrp2Iw0v/ema6QONYndQ==
Encrypted Text : [(array([ 7008,  8935, 53326, 41284, 55133,
 53195, 12864, 34598, 49047,
        12575,  1082, 33688, 20462, 59427, 28697, 30443, 4271
3, 19719,
        21854, 53300, 24143,  1231, 38940,  5541, 20994, 1569
2, 31362,
        39426, 18641, 59642, 35918,  7907], dtype=int64), arr
ay([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028,
53276,
```

In above screen source node 4 sending data to destination 15 by using neighbour node as 8 and in text area we can see all messages like Plain message, secret key, encrypted data and generated integrity code and then in below receiver screen will get decrypted data



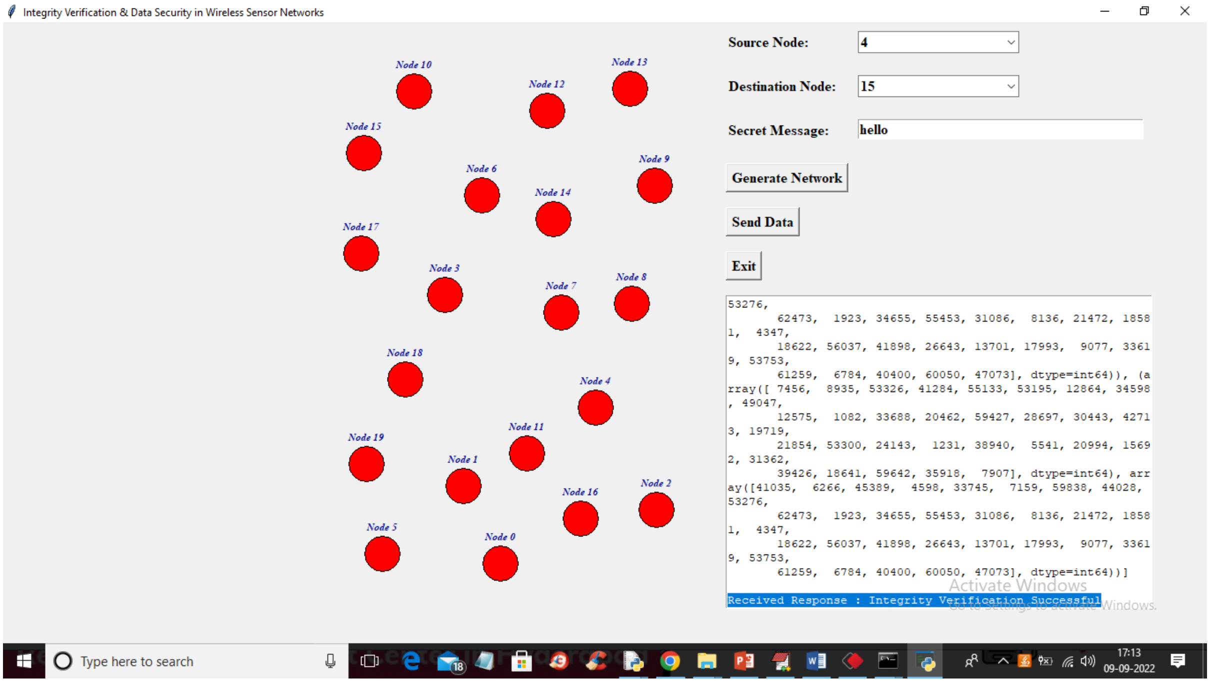
```
E:\manoj\Sep22\WSNVerification\Receiver>python Receiver.py
Receiver Node Started


Request received from Sender IP : 127.0.0.1 with port no : 62960

Source : 4 Destination : 15
Received Encrypted Data : [(array([ 7008,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 6816,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 7264,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 7264,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 7456,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64))]
Received Signature : ZQ86sUBKrSX1pLv2z0/c/tlzmufUzeiKa/KlnvPqmx0KisM7VZN93Yhbo/oJ6+zfDtIrp2Iw0v/ema6QONYndQ==
Decrypted Data : hello

Integrity Verification Successful
```

In above receiver screen we can see received signature, decrypted message and verification successful and will get acknowledgement in sender application like below screen



Source Node: 4

Destination Node: 15

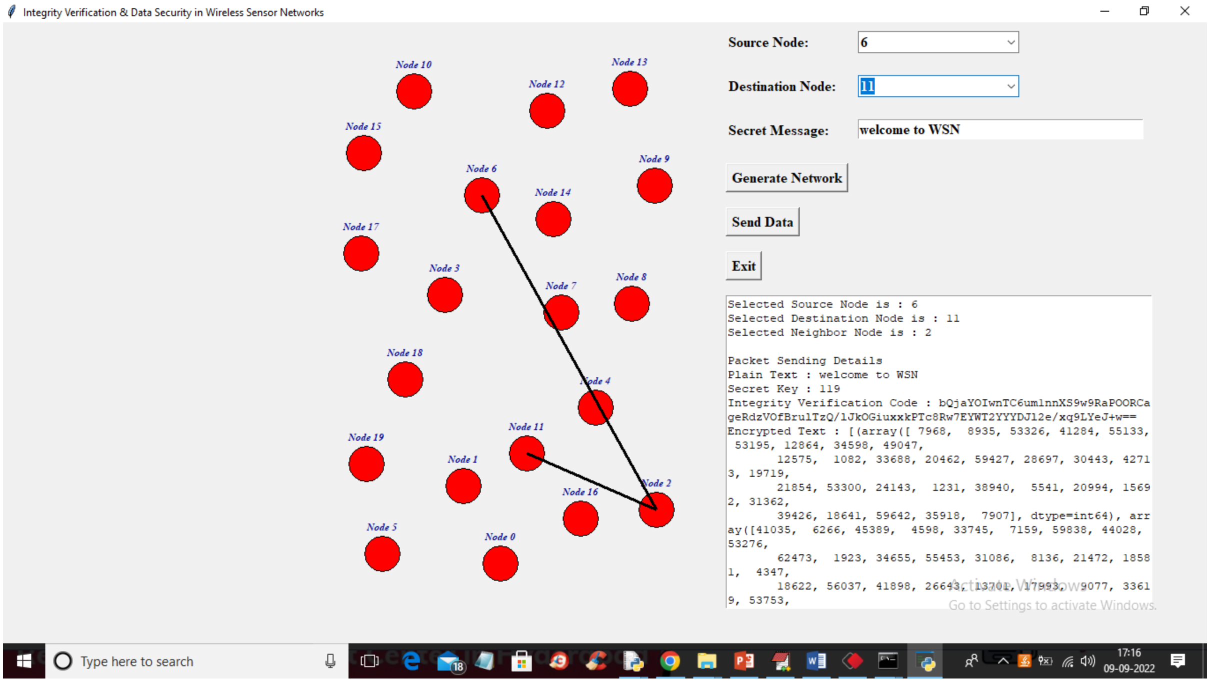
Secret Message: hello

```
53276,
        62473,  1923, 34655, 55453, 31086,  8136, 21472, 1858
1,  4347,
        18622, 56037, 41898, 26643, 13701, 17993,  9077, 3361
9, 53753,
        61259,  6784, 40400, 60050, 47073], dtype=int64)), (a
rray([ 7456,  8935, 53326, 41284, 55133, 53195, 12864, 34598
, 49047,
        12575,  1082, 33688, 20462, 59427, 28697, 30443, 4271
3, 19719,
        21854, 53300, 24143,  1231, 38940,  5541, 20994, 1569
2, 31362,
        39426, 18641, 59642, 35918,  7907], dtype=int64), arr
ay([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028,
53276,
        62473,  1923, 34655, 55453, 31086,  8136, 21472, 1858
1,  4347,
        18622, 56037, 41898, 26643, 13701, 17993,  9077, 3361
9, 53753,
        61259,  6784, 40400, 60050, 47073], dtype=int64))]

Received Response : Integrity Verification Successful
```

In above screen we can see Sender has got acknowledgement from Receiver that Integrity verification successful.

Similarly you can send and receive any number of message and Click on ‘Generate Network’ only one time and send data by selecting source and destination any number of time.



Source Node: 6

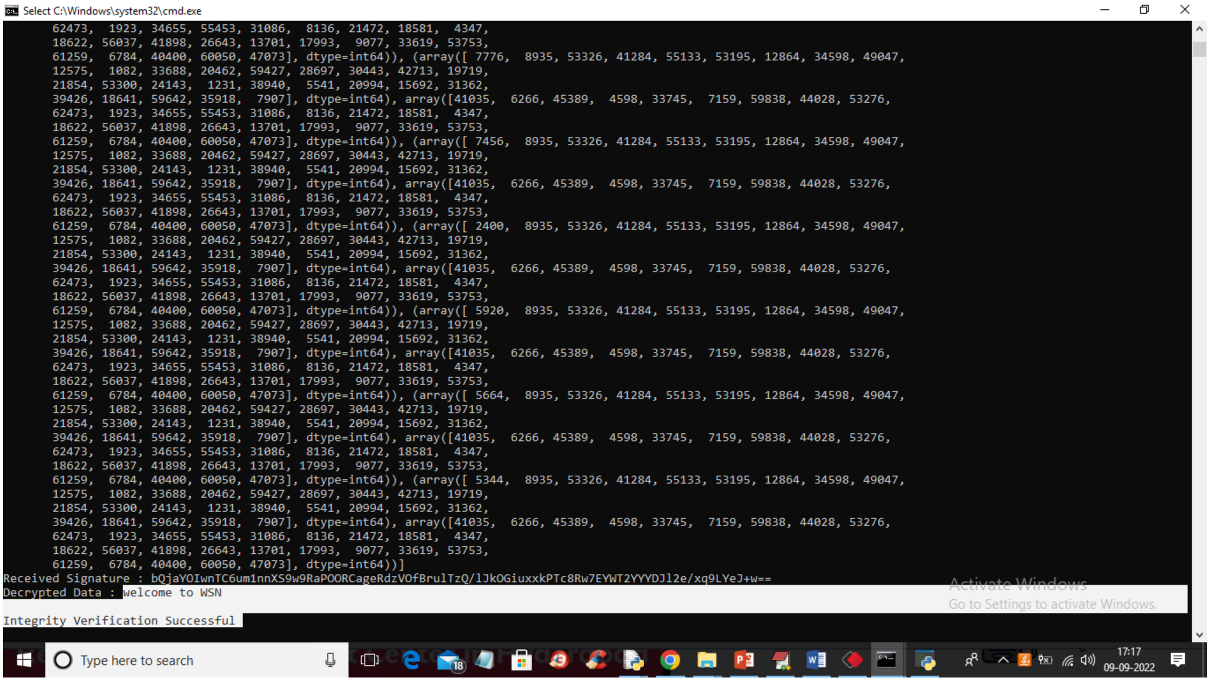
Destination Node: 11

Secret Message: welcome to WSN

```
Selected Source Node is : 6
Selected Destination Node is : 11
Selected Neighbor Node is : 2

Packet Sending Details
Plain Text : welcome to WSN
Secret Key : 119
Integrity Verification Code : bQjaYOIwnTC6umlnnXS9w9RaPOORCa
geRdzVOfBrulTzQ/lJkOGiuxxkPTc8Rw7EYWT2YYYDJl2e/xq9LYeJ+w==
Encrypted Text : [(array([ 7968,  8935, 53326, 41284, 55133,
 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 4271
3, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 1569
2, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), arr
ay([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028,
53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 1858
1,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 3361
9, 53753,
```

In above screen I am sending data from node 6 to 11 and Receiver will get below output



```
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 7776,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 7456,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 2400,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 5920,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 5664,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64)), (array([ 5344,  8935, 53326, 41284, 55133, 53195, 12864, 34598, 49047,
       12575,  1082, 33688, 20462, 59427, 28697, 30443, 42713, 19719,
       21854, 53300, 24143,  1231, 38940,  5541, 20994, 15692, 31362,
       39426, 18641, 59642, 35918,  7907], dtype=int64), array([41035,  6266, 45389,  4598, 33745,  7159, 59838, 44028, 53276,
       62473,  1923, 34655, 55453, 31086,  8136, 21472, 18581,  4347,
       18622, 56037, 41898, 26643, 13701, 17993,  9077, 33619, 53753,
       61259,  6784, 40400, 60050, 47073], dtype=int64))]
Received Signature : bQjaYOIwnTC6um1nnXS9w9RaPOORCageRdzVOfBrulTzQ/lJkOGiuxxkPTc8Rw7EYWT2YYYDJl2e/xq9LYeJ+w==
Decrypted Data : welcome to WSN

Integrity Verification Successful
```

In above screen we can see decrypted message with verification output

5. CONCLUSION

This project implemented the data integrity verification using Chaotic technique and data security using homomorphic encryption. In proposed technique, the system will send a verification hash code along with encrypted message to the receiver. Next, receiver will decrypt the received message and then it will re-generate the verification hash code. Finally, the verification will be successful if this received, and the generated hash code matches.

REFERENCES

[1] . Zhao, “On resilience and connectivity of secure wireless sensor networks under node capture attacks,” IEEE Trans. Inform. Forensics Secur., vol. 12, no. 3, pp. 557–571, Mar. 2017.

[2] Mishra and A. Turuk, “A comparative analysis of node replica detection schemes in wireless sensor networks,” J. Netw. Comput. Appl., vol. 61, pp. 21–32, 2016.

[3] Wald, Sequential Analysis. New York, NY, USA: Dover, 2013.

[4] S. Agrawal, M. L. Das, A. Mathuria, and S. Srivastava, “Program integrity verification for detecting node capture attack in wireless sensor network,” in Proc. Int. Conf. Inf. Syst. Secur., 2015, pp. 419–440.

[5] X. Jin, P. Putthapipat, D. Pan, N. Pissinou, and S. K. Makki, “Unpredictable software-based attestation solution for node compromise detection in mobile WSN,” in Proc. IEEE Globecom Workshop Adv. Commun. Netw., 2010, pp. 2059–2064.

[6] S. S. Desai and M. J. Nene, "Node-Level Trust Evaluation in Wireless Sensor Networks," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 8, pp. 2139-2152, Aug. 2019, doi: 10.1109/TIFS.2019.2894027.

[7] C. Miranda, G. Kaddoum, E. Bou-Harb, S. Garg and K. Kaur, "A Collaborative Security Framework for Software-Defined Wireless Sensor Networks," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2602-2615, 2020, doi: 10.1109/TIFS.2020.2973875.

[8] Cui, J., Shao, L., Zhong, H. et al. Data aggregation with end-to-end confidentiality and integrity for large-scale wireless sensor networks. Peer-to-Peer Netw. Appl. 11, 1022–1037 (2018). https://doi.org/10.1007/s12083-017-0581-5

[9] A. A. Agarkar, M. Karyakarte and H. Agrawal, "Post Quantum Security Solution for Data Aggregation in Wireless Sensor Networks," 2020 IEEE Wireless Communications and Networking Conference (WCNC), 2020, pp. 1-8, doi: 10.1109/WCNC45663.2020.9120843.

[10] X. Liu, J. Yu, F. Li, W. Lv, Y. Wang and X. Cheng, "Data Aggregation in Wireless Sensor Networks: From the Perspective of Security," in IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6495-6513, July 2020, doi: 10.1109/JIOT.2019.2957396.

[11] Raja Waseem Anwar, Anazida Zainal, Fatma Outay, Ansar Yasar, Saleem Iqbal, BTEM: Belief based trust evaluation mechanism for Wireless Sensor Networks, Future Generation Computer Systems, Volume 96, 2019, Pages 605-616, ISSN 0167-739X, https://doi.org/10.1016/j.future.2019.02.004.

[12] Fang, W., Zhang, W., Chen, W. et al. TMSRS: trust management-based secure routing scheme in industrial wireless sensor network with fog computing. Wireless Netw 26, 3169–3182 (2020). https://doi.org/10.1007/s11276-019-02129-w

[13] C. Lyu, X. Zhang, Z. Liu and C. -H. Chi, "Selective Authentication Based Geographic Opportunistic Routing in Wireless Sensor Networks for Internet of Things Against DoS Attacks," in IEEE Access, vol. 7, pp. 31068-31082, 2019, doi: 10.1109/ACCESS.2019.2902843.