Mitigating Replay Attack in Wireless Sensor Network Through Assortment of Packets

Vandana Sharma and Muzzammil Hussain

Abstract There are many attacks possible on wireless sensor network (WSN) and replay attack is a major one among them and, moreover, very easy to execute. A replay attack is carried out by continuously keeping track of the messages exchanged between entities and replayed later to either bring down the target entity or affect the performance of the target network. Many mechanisms have been designed to mitigate the replay attack in WSN, but most of the mechanisms are either complex or insecure. In this paper, we propose a mechanism to mitigate the replay attack in WSN. In the proposed work at each node, assorted value of a received packet is maintained in a table and the reply attack is detected or mitigated using their assorted value of the already received packets. The proposed mechanism was simulated and its performance evaluated and it was found that the proposed mechanism mitigates the replay attack, secures the network and takes less time for processing.

Keywords Replay attack · Sensor nodes · Hash · MD5 · Occurrence

1 Introduction

WSN was introduced first time in military and heavy industrial applications. Governments and universities began using WSN in many applications such as natural disaster prevention, weather stations and fire detection. With the increased use of the WSN mission in critical environments such as military and health-care applications, these environments need to be secured.

V. Sharma (✉) · M. Hussain
Department of Computer Science and Engineering, Central University of Rajasthan, Bandarsindri, Kishangarh, Ajmer 305817, Rajasthan, India
e-mail: 2014mtcse022@curaj.ac.in

M. Hussain
e-mail: mhussain@curaj.ac.in

S.C. Satapathy et al. (eds.), *Proceedings of the First International Conference on Computational Intelligence and Informatics*, Advances in Intelligent Systems and Computing 507, DOI 10.1007/978-981-10-2471-9_22

The sensor node is composed of the sensor, processor, transceiver, ADC, memory antenna and power generator. The sensor produces a considerable response whenever it detects any change in its external environment. The main component of the sensor node is the sensor; it senses the data and converts it from analog to digital through ADC. The data are then processed and the processed data are then sent through the transceiver to the base station. The transceiver does the task of both transmitting and receiving [1, 2]. For location finding and mobility handling, a location finding system and mobilizer are introduced. The power generator supplies power. WSNs are placed in a location where humans cannot reach easily, so that they should be provided with a sufficient amount of energy to power the system.

The major challenges of WSN are ad hoc deployment, dynamic nature and unattended operation. Due to the dynamic environment, sensor nodes should be developed so that they can cope with environmental challenges. Due to this, unattended operation sensor nodes should be configured so that they can adapt to the environment changes automatically. WSNs are deployed in places where humans cannot go physically, so physical security becomes a major concern. Confidentiality, authentication, integrity and availability are the major security requirements of WSN [2, 3].

WSNs have limited storage capacity and computational power. The battery of WSN should be used efficiently. All attacks on WSN are to deplete the resources of WSN. There are many attacks possible, such as selective forwarding, sinkhole, Sybil, impersonation and eavesdropping [4]. Replay attack can be carried out easily by keeping track of all the messages being sent between nodes and sending them back later, to waste the resources of the target node in processing the message. The target node gets drained and will be unable to perform its actual task, leading to denial-of-service attack [5, 6]. In this paper, we have studied the impact of replay attack and have proposed a mechanism to mitigate it.

2 Related Work

So far, mechanisms have been designed to detect and mitigate replay attack. Here, we summarize the effective mechanisms among them, but most of them suffer from problems such as security, complexity and synchronization.

2.1 Lamports Password Authentication

This method implements a onetime password, due to which the attacker cannot eavesdrop on messages exchanged. This method is implemented between the user and the server. A system which uses this method will never use the same password.

In this, a password table is used to verify the legality of the user's identity. In this mechanism, there is a risk of losing the password table; if it is used, the attacker can misuse it. To overcome the stolen table attack of the Lamport Password authentication scheme, nonce-based and timestamp-based schemes were introduced.

2.2 Nonce

In password-based authentication protocol, [7] the server sends a challenge or a random value to the client and the client will respond by sending h(c ‖ p), where h is the hash, c is the challenge and p is the password. ‖ denotes concatenation. The server checks the password in its database and whether it is correct; if yes, then the client is validated.

With a passive attack, the attacker eavesdrops, but does not alter the message. The attacker can see c and h(c ‖ p), so he can use this cluster to create passwords until a match is found. Random challenge is used to avoid this attack.

If the attacker is active, then he can change the messages; the attacker can send his own challenge c' and wait for the client's response. The client will send the response h(c' ‖ p). Using the same challenge c', he can get the precomputed table. Now, the attacker can attack several passwords. A client nonce avoids this by the following:

The server sends a random challenge. The client chooses a nonce n.
The client sends n ‖ h(n ‖ c ‖ p).

The server re-computes h(c ‖ n ‖ p) and sees if this value matches the one the client sends.

2.3 Timestamps

It is a method to ensure the freshness of the message. This is a very old method to find out that the message which is received is original or replayed. In this, every message is sent along with a timestamp [8]. The receiver after receiving the message checks whether the timestamp is within the acceptable range; if yes, then the message is accepted otherwise dropped. Through Network Time Protocol every computer maintains accurate time.

To implement this method, clock synchronization of the two entities is a must. Synchronization is required to maintain the accuracy and precision of the timestamp. Here, we need to make a list of old timestamps, large storage is required and due verification overhead is suffered.

2.4 Sequence Number

Sequence number is assigned to be monotonically either increasing or decreasing to each transmitted message. If a message is replayed, it will have a very small or very old sequence number and can be discarded [9]. In this mechanism, forced replayed messages cannot be detected and also it is difficult to maintain the sequence number.

2.5 Receiver Authentication Protocol

RAP [7] can be used for two purposes, detection and prevention. Detection is a scheme which aims to detect an intruder that replays beacons without preventing it from doing so. In the prevention mode, the challenge–response message exchange takes place before data transmission. In this, the sender transmits the data only when the receiver is authenticated. Energy efficiency is a major requirement; so in normal condition, detection is done and the system switches to the expensive prevention mode only if required.

2.5.1 Detection Mode

RAP-D aims to detect the replayed packets. As we can see in Fig. 1, if a sender node A wants to transmit data to receiver B, B broadcasts a beacon and A answers backs with data and a challenge Cp. On the following beacon, B acknowledges the reception of data packets and sends encrypted version of challenge Cp using shared

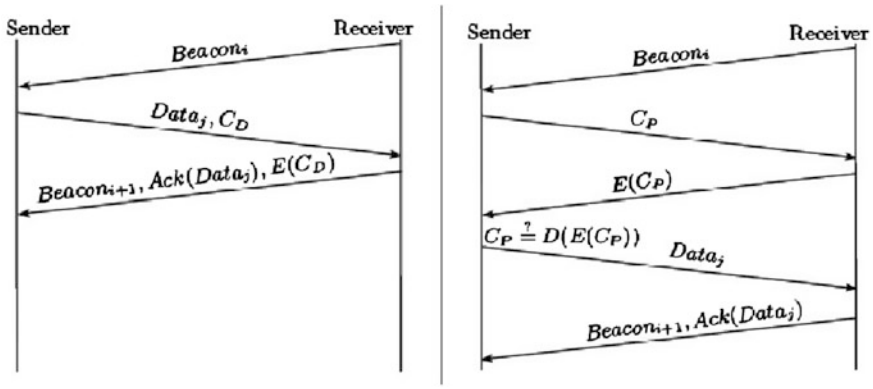

Fig. 1 Detection of the replayed beacon shown in the first figure and prevention of beacon replayed shown in the second figure

key Krap. B validates the response by decrypting it and comparing it with the original value. If they are the same, then the message is not replayed.

2.5.2 Prevention Mode

RAP-P aims to prevent the beacon replay at the cost of increased overhead. It can be seen in Fig. 1. In this, data is not sent right after the beacon. A sends a challenge Cp and waits for the encrypted challenge from B. Only if the received value is correct, then the data is exchanged. This is more costly because two more messages are being sent.

3 The Proposed Algorithm

In this work, we have designed a mechanism to mitigate replay attack. In the proposed protocol, for each received message, the node calculates its hash value and stores it in the table. The computed hash is searched in the occurrence table; if not found, then the message is treated as fresh and it is entered into the occurrence table with occurrence set to one. Otherwise, the message is replayed once and is discarded by the node.

Algorithm

```
Create table()
{
   Struct table
   {      char hp;
          Int occur;
    }
Struct table t[100];
}

For each received packet M at any node
For I from 1 till n
   If (table[1].id == hp &&  table[1].occur > 0) then
        Print " Packet is replayed"
   Else
         Make entry in table
         Table[n+1].id = hp;
         Table[n+1].occur = 1;
         Print "packet is fresh"
         Break;
End
```

where n is the maximum number of entries in the table, Table[i].id is the hash of the ith entry and hp is the hash value.

Whenever a message is received at any node, it is checked that a table is created or not; if yes, then the hash of the received message is compared with the already existing hashes. If a match is found, then the message is declared as a replayed message; otherwise, it is a new message and entry is created in the table for hp and occur. Here, hp is hash of the received message and the occur value is set to one. Loop 'n' is the total number of entries in the table. This loop will run for every message which is received. The table is a structure, which consists of two members table[i].id for hash of message and table[i].occur for occurrence value.

If table[i].id is equal to the recently calculated hash and table[i].occur > 0, then the message is replayed; otherwise, a new entry is created for the hash and occur value and the packet is declared as fresh.

4 Implementation

The proposed protocol was implemented in Python database. For enabling communication, three entities are defined, two clients and one server. Clients communicate through the server. The proposed protocol is implemented at the server, while replayed messages are discarded by the server and fresh messages forwarded to clients. The server is defined as a local host with IP address 127.0.0.1 at port 5000. Clients are at port 0. Socket API is used for communication between client and server.

$$\text{ipc s} = \text{socket.socket}(\text{socket.AF INET}, \text{socket.SOCK DGRAM}).$$

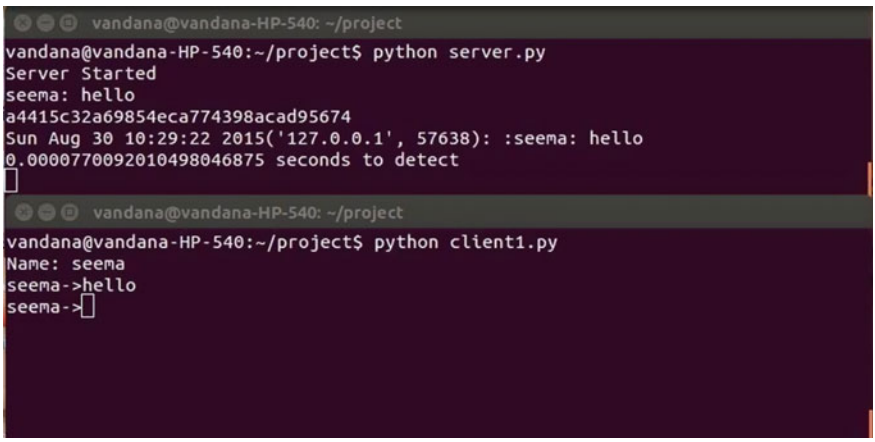

Fig. 2 client1 sending a hello message to client2 through the server

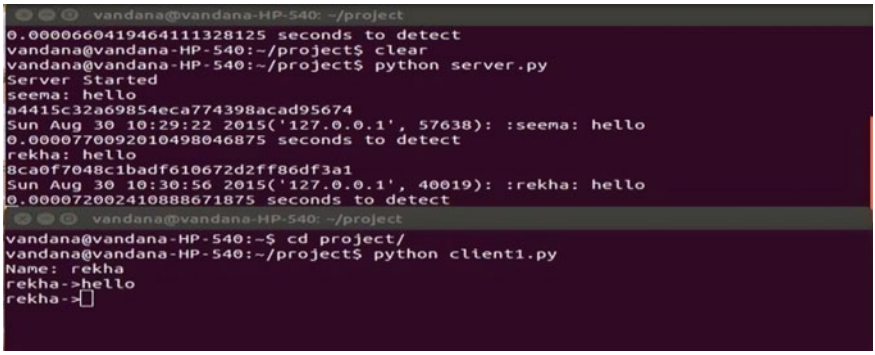

Fig. 3 client2 sending a hello message to client1 through the server

The messages are exchanged between communicating entities as datagram. The hash of the received message is calculated using MD5 as g = hashlib.md5(str (data))hashdigest().

5 Result and Analysis

The proposed algorithm was simulated in Python. One server and two clients (client1 and client2) were created. client 1 will send a hello message to client 2 and client2 responds with a hello message. If client1 sends a hello message again, the

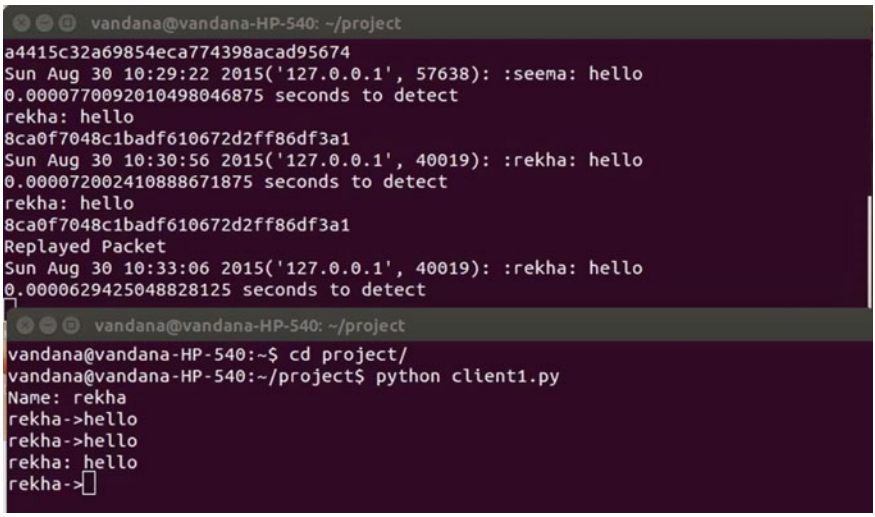

Fig. 4 The server detecting a replayed hello message sent by client1

server detects it as a replayed one. The proposed mechanism is simple as the nodes just need to compute the hash of a received packet and compare it with the existing values in the table. The mechanism is secure, as it is impossible for any node to mislead the receiver node by resending packet as a fresh one by changing a few parameters because the hash is computed for the whole packet (Figs. 2, 3 and 4).

6 Performance Comparison of Bloom Filter with the Proposed Protocol

The performance of the proposed protocol is compared to that of the existing protocol (Bloom filters). It has been observed that the proposed protocol has a less number of hash computations and the probability of detecting a replayed packet is the same. However, the proposed protocol takes less time than that taken by Bloom

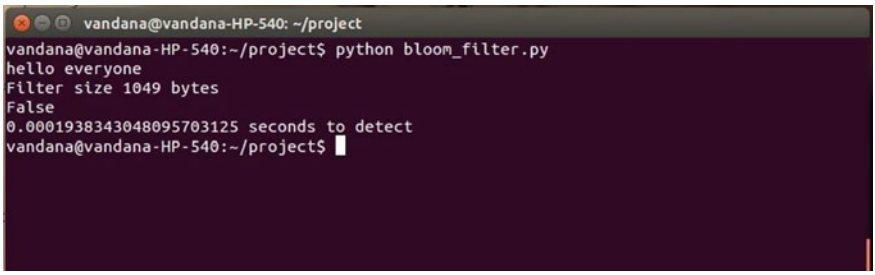

Fig. 5 Time taken to detect the replay message through the Bloom filter is 0.0001938343 s for **a** particular input

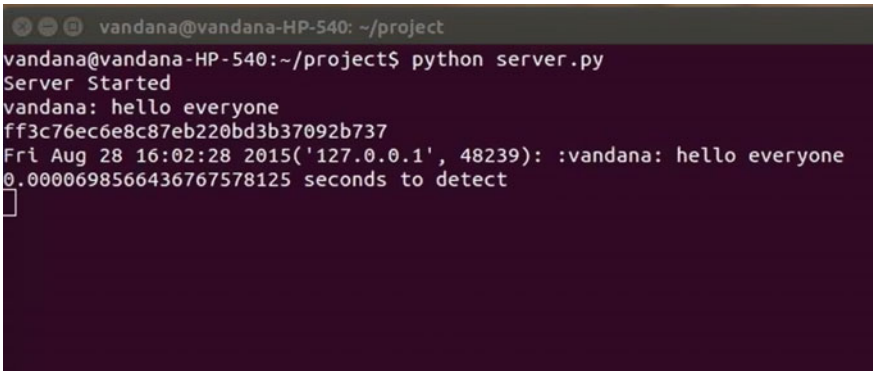

Fig. 6 Time taken to detect the replay message through the proposed protocol is 0.000069856643 s for the same input as in the Bloom filter

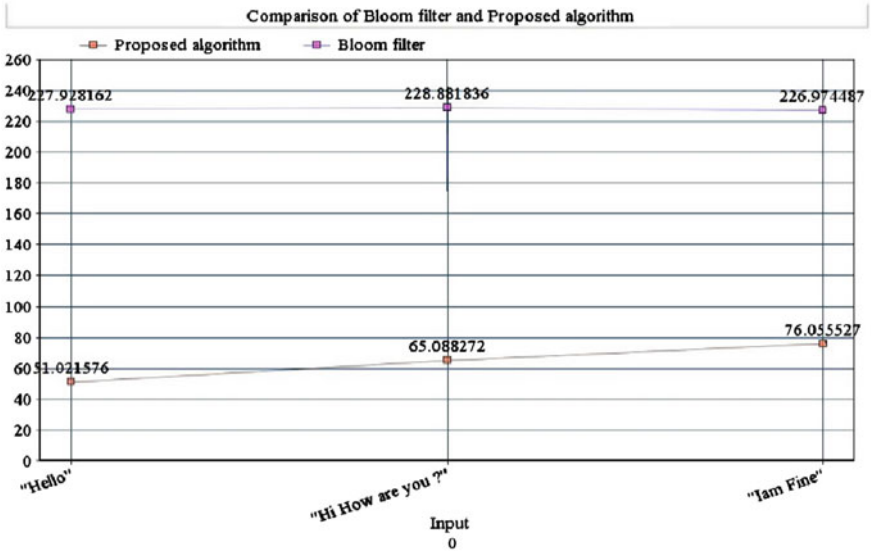

Fig. 7 The comparison of the time taken by the Bloom filter and the proposed algorithm for various inputs. Inputs are mapped in the X-axis and time in milliseconds is on the Y-axis. It shows that the time taken by the proposed algorithm is very less compared to that by the Bloom filter

filters mechanism. The computation time of the proposed protocol was found to be 0.0006985 s and that of the bloom filters mechanism 0.0001950 s for the same and under a similar environment (Figs. 5, 6 and 7).

7 Conclusion

Security is a major issue in WSN. The network has to be protected against various possible attacks so as to extend the lifetime of sensor nodes and also to avoid its malfunctioning. Replay attack is very common in WSN and may lead to many more attacks like DoS. Hence, it is very much needed to design a mechanism to mitigate any replay attack in WSN.

We have designed an algorithm to detect and mitigate any replay attack. The receiving entity calculates the hash of the received packet and stores in its database along with a parameter occurrence. Any packet with occurrence one is detected as replayed. The proposed algorithm is implemented in Python and its performance is studied and verified. It has been found that the proposed protocol is successful in mitigating the replay attack in WSN. The time taken by the proposed protocol is found to be less then that compared to the existing mechanism.

References

1. Syverson, Paul.: A taxonomy of replay attacks [cryptographic protocols] Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings. IEEE, 1994.
2. Pathan, Al-Sakib Khan, Hyung-Woo Lee, and Choong Seon Hong.: Security in wireless sensor networks: issues and challenges. Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference. Vol. 2. IEEE, 2006.
3. Karlof, Chris, and David Wagner.: Secure routing in wireless sensor networks: Attacks and countermeasures. Ad hoc networks 1.2 (2003): 293–315.
4. Jesudoss, A., and N. Subramaniam.: A survey on authentication attacks and countermeasures in distributed environment. IJCSE, vol 5.2 (2014).
5. Raymond, David R., and Scott F. Midkiff.: Denial-of-service in wireless sensor networks: Attacks and defenses. Pervasive Computing, IEEE 7.1 (2008): 74–81.
6. Raymond, David R., et al.: Effects of denial-of-sleep attacks on wireless sensor network MAC protocols. Vehicular Technology, IEEE Transactions on 58.1 (2009): 367–380.
7. Di Mauro, Alessio, et al.: Detecting and preventing beacon replay attacks in receiver-initiated MAC protocols for energy efficient WSNs. Secure IT Systems. Springer Berlin Heidelberg, 2013. 1–16.
8. Baayer, Aziz, Nourddine Enneya, and Mohammed Elkoutbi.: Enhanced timestamp discrepancy to limit impact of replay attacks in manets. (2012).
9. Sung-Ming, Yen, and Liao Kuo-Hong.: Shared authentication token secure against replay and weak key attacks. Information Processing Letters 62.2 (1997): 77–80.