CHAPTER-4
SECURITY ATTACKS ON
HETEROGENEOUS WIRELESS SENSOR NETWORKS

4.1. HETEROGENEOUS WSNs

A heterogeneous wireless sensor network (WSN) consists of several different types of sensor nodes (SNs). Various applications supporting different tasks, e.g., event detection, localization, and monitoring may run on these specialized sensor nodes. In addition, new applications have to be deployed as well as new configurations and bug fixes have to be applied during the lifetime. In a network with thousands of nodes, this is a very complex task. A heterogeneous node has more complex processor and memory so that they can perform sophisticated tasks compared to a normal node. A heterogeneous node possesses high bandwidth and long distant transceiver than a normal node proving reliable transmission.

4.2. TYPES OF HETEROGENEOUS RESOURCES

There are three common types of resource heterogeneity in sensor node:

4.2.1. Computational Heterogeneity

Computational heterogeneity means that the heterogeneous node has a more powerful microprocessor and more memory than the normal node. With the powerful computational resources, the
heterogeneous nodes can provide complex data processing and longer term storage.

4.2.2. Link Heterogeneity

Link heterogeneity means that the heterogeneous node has high bandwidth and long-distance network transceiver than the normal node. It can provide more reliable data transmission.

4.2.3. Energy Heterogeneity

Energy heterogeneity means that the heterogeneous node is line powered, or its battery is replaceable. Among above three types of resource heterogeneity, the most important heterogeneity is the energy heterogeneity because both computational heterogeneity and link heterogeneity will consume more energy resource. If there is no energy heterogeneity, computational heterogeneity and link heterogeneity will bring negative impact to the whole sensor network, i.e., decreasing the network lifetime.

A heterogeneous node is line powered (its battery is replaceable). The heterogeneous WSN consists of different types of sensors with different sensing and transmission range. So while selecting the sensor nodes for intrusion detection, we need to consider these inequality of sensing and transmission range. For example, if two nodes have different transmission range it is better to select the one whose transmission range is higher. In this paper, we are considering N types of sensors. Here the sensing range and
transmission range is high for Type 1 compared to Type2 and so on. The sensors are uniformly and independently deployed in an area \( A=L \times L \).

### 4.3. COMPARATIVE STUDY OF HETEROGENEOUS WSNs AND HOMOGENEOUS WSNs

In homogeneous networks, all the sensor nodes are identical in terms of battery energy and hardware complexity. Heterogeneous networks achieve the former and the homogeneous networks achieve the latter. In homogeneous network, single (uniform) platform is used for research group and all nodes in the network share the same functionality where as in heterogeneous network all the nodes treated differently. In the real world, the assumption of homogeneous sensors may not be practical because sensing applications may require heterogeneous sensors in terms of their sensing and communication capabilities in order to enhance network reliability and extend network lifetime. Also, even if the sensors are equipped with identical hardware, they may not always have the same communication and sensing models. In fact, at the manufacturing stage, there is no guarantee that two sensors using the same platform have exactly the same physical properties. This taxonomy focuses on heterogeneity at the designing stage, when sensors are designed to have non identical capabilities to meet the specific needs of sensing applications.

In the heterogeneous wireless sensor network, the average energy consumption for forwarding a packet from the normal nodes to
the sink in heterogeneous sensor networks will be much less than the energy consumed in homogeneous sensor network.

4.4. INTRODUCTION TO SECURITY ATTACKS

Wireless sensor networks (WSNs) have become a hot research topic in recent years. Applications include military, rescue, environment monitoring, and smart homes. A WSN is composed of hundreds or even thousands of small, cheap sensors nodes which communicate with one another wirelessly. Sensor nodes typically do not have very much computational power, limiting the kinds of networking protocols and security mechanisms they can employ. Because WSNs are composed of so many nodes, which may be deployed in a hostile environment, replacing batteries is not feasible. Sensor nodes must there for survive on the small amount of energy in the batteries they are deployed with (typically about 6 amp-hours [5]). This creates a need to conserve energy. Because of the wireless nature of WSNs, security is a fairly difficult issue. Adversaries can easily listen to all the traffic and inject their own, especially if the WSN is deployed in a hostile environment. It is also important that the WSN be robust to losing some of the sensor nodes, because it can be very easy for an adversary to capture any given node.

The general network topology is a dense collection of nodes, randomly distributed over some geographic area. Traffic typically goes from all the sensor nodes to a single sink, called the base station (BS), or broadcast traffic. A lot of work is currently being done on routing
protocols, and not all of the details are figured out and agreed upon but, in general, routing is multi-hop like an ad-hoc wireless network. Cluster-based routing is a popular idea, because it is possible to exploit the fact that nearby nodes has highly correlated data [28]. In cluster-based routing, the network is divided up into clusters, which consist of a cluster head (CH) and member nodes (MNs). The MNs send their data to the CH, which aggregates the data before sending it out of the cluster toward the base station.

In the present study, I will assume that the WSN is in a hostile environment, where the deployers have no physical contact with the nodes, but attackers may. An attacker is someone who tries to disturb the functionality of the network in any of the ways described in the next session. I will also talk about “intrusion detection”. This is defined as identifying an intruder, which is an attacker who has gained control of a node, or injected falsified or repeated packets into the network. This is not to be confused with other “intrusion detection” systems using WSNs, which monitor a physical environment, looking for intruders using a WSN for sensing and collecting information.

In the present study, it will explain the special circumstances presented by WSNs, summarize work that has been done in security for WSNs and propose a new intrusion detection method. In the next section, we classified the types of attacks on WSNs. In the next coming sections, we explained a few of the protocols that can be used
for WSN security, discussed how clusters can be used for security purposes.

**4.4.1. Classification of Attacks**

There are four aspects of a wireless sensor network that security must protect:

1. Confidentiality
2. Data Integrity
3. Service Availability
4. Energy

The first three are addressed by security systems in wired networks and non-energy-constrained wireless networks, but the fourth is unique to the sensor network application. It is thus classified the attacks that can be launched by which of these aspects they attack. In this section It is described the kinds of attacks in each of these categories and the ways in which the nature of the wireless network causes trouble.

**A. Stealing Data (Confidentiality)**

When we think of electronic security, this is the first kind of attack that comes to mind. We want to be able to send messages without enemies being able to figure out the contents. Because of the wireless nature of the WSN, it is easy for an attacker to listen in on all the messages sent in the network, so to maintain confidentiality, the network must encrypt all the messages.
One of the biggest ideas in encryption today is public-key encryption. This is very powerful because it allows one to receive encrypted messages without even sharing a secret key with the sender. But this asymmetry comes at a cost. RSA public key encryption involves exponentiating the message, which can be quite computationally expensive, and is not really feasible in WSNs, where the nodes typically are not capable of doing such computations in a time- and energy-efficient way. But symmetric key encryption mechanisms pose the problem that if any node is apprehended by the attacker, he can look in its memory to find the key and effectively be able to masquerade as any other node (compromising the data integrity) and listen in on any other conversation.

**B. Altering/Generating False Data (Data Integrity)**

Because sensor networks are used to monitor some environment, data integrity is even more important than confidentiality. This is because applications may include tracking objects that physically move through the environment and since attackers can typically see the physical environment, the only thing they could gain from listening in on the data is a sense of where the sensors are located. On the other hand, if they are able to alter to make the data collected by the WSN incomplete or incorrect, the deployer of the WSN will not know what is really going on in the environment he is trying to monitor.
In other networks, the same asymmetric key system that is used for encryption can be used for digital signatures, but this requires a lot of additional overhead. The signature may consist of a lot of additional bytes of data added on to a transmission (which takes additional energy), and verifying the signature can be very computationally expensive. Clearly, different techniques are needed for WSNs.

C. Attacks on Service Availability

This class of attacks is not at all concerned with the actual data that is being sent. Rather, the goal is to make the network not function properly. This can be done by sending bogus routing information (for example advertising a route that does not exist). It can also be done by flooding the network with packets (denial of service attack), or even jamming the frequency at the physical layer.

Another interesting type of attack is homing. In a homing attack, the attacker looks at network traffic to deduce the geographic location of critical nodes, such as cluster heads or neighbors of the base station. The attacker can then physically disable these nodes. This leads to another type of attack: the “black hole attack”. In a “black hole” attack, the attacker compromises all the neighbors of the base station, making it effectively a black hole. A final kind of attack on service availability is a de-synchronization attack, where the attacker tries to disrupt a transport-layer connection, by forging packets from either side [16].
D. Denial of Sleep Attacks (Energy)

The constrained energy of WSNs adds a new element that can greatly complicate security issues. Because there is a limited amount of energy available and no way to replenish it, it is not sufficient to make sure that bad data is not used. We need to make sure that we do not waste energy listening to or re-transmitting bad packets. This introduces a whole new set of possible attacks. These include constantly sending RTS packets to stop nodes from going to a low power “sleep” state, sending falsified or repeated packets so that nodes waste energy re-transmitting them, or draining the power of a node by forcing it to do excessive computations [71].

4.4.2. Protocols Used for WSN Security

A. SPINS

Many of the confidentiality and data integrity issues can be handled by SPINS [15]. SPINS is a collection of protocols for sensor networks. The key security components are SNEP and μTESLA. SNEP provides a lot of key security features. It provides confidentiality and data integrity for pair wise connections as well as weak freshness. Freshness means that old packets cannot be repeated by an adversary to create confusion and waste energy. Weak freshness means that there are no delay guarantees, but packets cannot be repeated or re-ordered. In SNEP, each pair of nodes shares a pair-wise key. This key is used in DES in cipher block chaining (CBC) mode. The cipher block chain provides semantic security (meaning that the same message
string will not always encrypt to the same cipher string) through the use of an initialization vector (IV). Rather than sending this IV in the clear along with a message, the IV comes from a shared counter. This alleviates the need to send unnecessary bits. The counter also provides data freshness, because since it is incremented with each transmission, a previous transmission cannot be repeated, and the correct ordering is evident. In addition to being encrypted, messages are also authenticated in SNEP through the use of a message authentication code (MAC), which is a function that takes two arguments and maps them to an 8 byte number. The arguments used are the pairwise key and the concatenation of the count and the encrypted message. Because all of these are available to the receiver, it can calculate the function to verify the signature.

SNEP is very good because it provides all of this security for an overhead of merely 8 bytes per message. While SNEP provides pairwise authentication, it cannot provide authentication for broadcasts, because if the key is shared among several nodes, compromising any of them would allow the attacker to masquerade as any node in the group. Broadcast authentication is therefore handles using μTESLA. Because of the nature of broadcast, asymmetry of information is very important. The sender (typically the base station or cluster head) must be able to generate a signature that the other nodes can verify, but not create on their own. μTESLA accomplishes this using a one-way
function. It generates a sequence of keys to be used in the MAC in the following way:

\[ K_n = F(K_{n+1}). \]

It is important that the keys are used in reverse order, because this assures that a key cannot be used to predict future keys (\( F(\cdot) \) is one-way). In addition, the keys are disclosed periodically, rather than with each packet, because this saves the unnecessary overhead. An example of \( \mu \)TESLA can be seen in fig. 4.1. In fig. 4.1 we can see that if a node can buffer P1 through P5, it only needs to hear K4 to be able to authenticate all of the previous packets. Having each key represent a given time frame rather than a given packet assures that even if a packet is missed, all received packets can be authenticated.

![Fig. 4.1: An example of \( \mu \)TESLA](image)

**B. Maintaining Service Availability**

Because of the nature of attacks to service availability, most can be dealt with without specific protocols to defend against them. Bogus routing information will be avoided if the packets used to determine routing are properly authenticated. Homing can be limited through the use of encryption to conceal the contents of the packets. De-
synchronization is avoided by handling the security at the MAC layer. One of the more dangerous and hard-to-avoid attacks is physical layer jamming. The response at the physical layer would be to use frequency hopping or CDMA spread-spectrum modulations [84, 68]. Forward error correction may also be included at higher layers to add redundancy, so that short bursts of noise can be compensated for [14]. However, if the attacker has unlimited power, which may be the case, this cannot be avoided, since the attacker can simply increase the power and duration of the jamming.

4.4.3. Cluster-Based Security

As we have seen in the G-MAC example, clusters can provide major advantages in sensor network security. In the case of G-MAC, we let the GS be the CH. The CH can also monitor the traffic coming from each MN and figure out if any of them have been compromised. It can then blacklist these nodes, isolating them from the network. In case a CH is compromised, MNs must also have the ability to decommission the CH if there are enough MNs that agree to do so [28]. This will defend against homing attacks. It is critical that several nodes agree to decommission the cluster head, because if only a few nodes are compromised, they should not be able to take down the cluster head. When a node is removed, its transmissions will be ignored, and nothing will be sent to the node. However, it will still be able to hear and understand broadcast traffic. This can be fixed with a slight modification to µTESLA in which the function $F(\cdot)$ is not fixed at
deployment time, but is rather a function that takes two arguments: a cluster key and key for the broadcast Kn. We can then generate keys using

$$K_n = F(k \oplus K_{n+1}).$$

When a node is removed, the CH can generate a new, and inform each of the remaining nodes individually. An analysis of this protocol modification may be included in future work. When a node is monitoring another node, there are two ways in which it can detect misbehavior: anomaly detection and signature detection [39]. In anomaly detection, the monitoring node looks for deviations from typical behavior. This has a high probability of false alarm, so it is not often used [39]. Signature detection, on the other hand, looks for particular types of misbehavior. This leaves it susceptible to new creative types of attacks, but there are not really a lot of new different actions a misbehaving sensor node could take. The typical actions in this application would be dropping packets, duplicating packets, or causing collisions [28].

The simplest idea for intrusion detection would be to have all nodes monitor each other in promiscuous mode, meaning they listen in on all transmissions, even if they are not the recipient, but this wastes way too much energy. The other extreme would be to only check the packets at the end (base station or its neighbors), but by waiting to check the packet, we run the risk of transmitting a bad
packet far too many times, consuming valuable resources. The packet should be checking only a few times, and early in the path. [39] Suggests that this should occur at the cluster head. All packets should be first sent to the cluster head, which checks their authenticity before forwarding them out to the rest of the network. But a malicious node can avoid this by neglecting to send its packet to the cluster head first, instead opting to send it out to another cluster. To compensate for this, every node should have a probability, p, of sending a packet to its cluster head to be checked. The higher the p, the more energy consumed in checking, and the longer the route is made, but the more likely we are to catch malicious packets.