5.4.2 Intrusion Detection in WSNs

1. Brutch et al. presented three different architectures for intrusion detection. The first is a stand-alone architecture. In this case, each node functions as an independent intrusion detection system and is responsible for detecting attacks directed towards it.
2. The nodes do not exchange and intrusion data and no cooperative detection mechanisms are deployed. The second architecture is a distributed and cooperative architecture. In this architecture, an intrusion detection agent is deployed on each node. While the local agents are responsible for detecting local attacks on the nodes, they also cooperate among themselves by exchanging intrusion related data to detect global intrusion attempts. The third architecture proposed is a hierarchical architecture. This is suitable for a multi-layered WSN, where the network is divided into clusters with the cluster-head node being responsible for routing within a cluster. The multi-layered networks are primarily used for event correlation.
3. Zhu et al. proposed an interleaved hop-by-hop authentication (IHOP) scheme. IHOP guarantees that the base station will detect any injected false data packets when no more than a certain number t of nodes are compromised.
4. The sensor network is organized in a cluster-based hierarchy. Each cluster-head builds a route to the base station and each intermediate node has an upper associate node and a lower associate node that is t + 1 hops away.
5. IHOP uses a number of shared keys namely, (i) every node shares a master key with the base station, (ii) each node knows its one-hop neighbors and has established a pair-wise key with each of them, (iii) a

node can establish a pair- wise key with another node that is multiple hops away if needed.

6. Further, IHOP also assumes that the base station has a mechanism to authenticate broadcast messages, e.g., TESLA. A cluster-head collects information from its members and sends a report to the base station only when at least t + 1 sensors observer the same result. Meanwhile a cluster-head also collects the MACs from detecting nodes.
7. Each detecting node sends two MACs to the cluster-head: a MAC using the key shared with the base station, referred to as the individual MAC, and a MAC using the key shared with its upper associate nodes, referred to as the pair-wise MAC. The cluster-head then compresses the t + 1 individual MACs by XORing them to reduce the size of the report.
8. However, the pair-wise MACs are not compressed for transmission. If they were, a node replaying the message would not be able to extract the pair-wise MACs and a compressed MAC for the base station. When an intermediate node receives a report, it verifies the MAC of its lower associate node. If it fails, the report is eliminated.
9. Otherwise, it removes the MAC, generates a new MAC using its upper associate node pair-wise key, and appends it to the report. However, the pair-wise MACs are not compromised for transmission. If they were, a node relaying the message would not be able to extract the pair-wise MACs of interest to it. Thus, a legitimate report includes t + 1 pair-wise MACs and a compressed MAC for the base station.
10. When an intermediate node receives a report, it verifies the MAC of its lower associate node. If it fails, the report is eliminated.
11. Otherwise, it removes the MAC, generates a new MAC using its upper associate node pair-wise key, and appends it to the report. IHOP ensures that the base station can detect false data packets when no more than

t nodes are compromised.

5.5 Software based Anti-Hamper Technique

1. Tampering another DOS attack in physical layer is tampering. By physical access an attacker can extract sensitive information such as cryptographic keys or other data on the node. A compromised node creates, which the attacker controls by altering or replacing node. Vulnerability of this attack is logical. One defense to this attack involves tamper-proofing the node's physical package.
2. A sensor network is usually built with a large number of small devices, each of which has limited battery energy, memory, computation, and communication capacities. Such sensor networks can be used for various critical applications such as the safeguarding of and early warning systems for the physical infrastructure that includes buildings, transportation systems, water supply systems, waste treatment systems, power generation and transmission, and communication systems.
3. Despite the critical role in their intended applications, sensor networks are vulnerable to various security attacks, especially because they are deployed in a hostile and/or harsh environment. In such an environment, a captured sensor may be reverse-engineered, modified, and abused by the adversary. That is, the adversary can,
   i) Acquire (via analysis of the sensor memory) detailed knowledge of what the sensor's program is supposed to do and what the master secret is.
   ii) Modify the program with a malicious code.

iii) Produce and deploy multiple copies of the manipulated sensor device in the network. This is a serious problem, as sensor devices, once compromised, can subvert the entire network, for example, blocking nodes within its communication range from receiving and/or sending/relaying any information.

4. Consequently, it is essential to make a sensor device tamper-proof.

5. Traditionally, the tamper-proofing of programs or a master secret relies on tamper-resistant hardware. However, this hardware-based protection will likely fail to provide acceptable security and efficiency because,

   i) Strong tamper-resistance is too expensive to be implemented in resource- limited sensor devices.

   ii) The tamper-resistant hardware itself is not always absolutely safe due to various tampering techniques such as reverse-engineering on chips, micro-probing, glitch and power analysis, and cipher instruction search attacks.

6. Existing approaches to generating tamper-resistant programs without hardware support can be classified as, code obfuscation that transforms the executable code to make analysis/modification difficult, result checking that examines the validity of intermediate results produced by the program, self-decrypting programs that store the encrypted executables and decrypt them before execution, and self-checking that embeds, in programs, codes for hash computation as well as correct hash values to be invoked to verify the integrity of the program under execution.

7. However, for the following reasons, these approaches are unsuitable for sensor networks where a program runs on a slow, less-capable CPU in each sensor device. First, in the case of code obfuscation, it becomes

easier to tamper with the program code as the code size in low-cost sensor device s shrinks, let alone the theoretical difficulty of obfuscation. Moreover, just making it difficult to tamper with program code is not sufficient as it cannot protect against "determined" attackers.

8. Second, techniques based on result-checking or self-decryption are too "expensive" to be employed in resource-limited sensor devices because they continuously incurs the overhead of verification or decryption, shortening the sensor's battery lifetime and degrading the network throughput.

9. Third, the security of self-decrypting programs can be easily broken unless the decryption routines are protected from reverse-engineering, for example, by means of hardware. Likewise, self-checking techniques become defenseless once the hash computation code and/or the hash values have been identified/analyzed by the adversary.

10. To defend the sensor network against the above-mentioned attacks, the following security conditions should be met,
    i) The program residing in a sensor is not modified (integrity) and, optionally,
    ii) The sensor identifier (ID) is unique in a network (uniqueness).

11. The second condition is needed only if certain services rely on unique IDs for their proper operation as the adversary may deploy cloned sensors to sabotage the services.

12. However, these conditions are difficult to meet due mainly to the usually hostile operational environment, as well as the very large size of sensor networks, under which it is easy for an adversary to capture and compromise sensors. Therefore, it needs an approach that creates a network of mutually trusted sensors, that is, each sensor can trust that the rest of the network has not been tampered with.

13. To achieve this, it is required that each sensor register itself with a dedicated server after verification of its program.