Chapter 4
Security in WSNs

Security in wireless sensor networks (WSNs) is centered on at least the following six fundamental requirements, namely: authentication, confidentiality, integrity, reliability, availability and data freshness [4, 39, 52, 62]. In this chapter, we describe these requirements, the different kinds of attacks that aim to compromise these requirements (and hence the security of a WSN) and the defense mechanisms that can be employed to overcome these attacks. While many security concepts hold true for different kinds of computer networks, WSNs have certain properties that make them particularly susceptible to attack relative to the traditional computer networks (e.g., wired networks). These properties do not only make possible a range of attacks not seen in regular computer networks, but also make a number of conventional defenses unsuitable for WSNs. We first describe examples of these unique properties and how they give rise to WSN-specific security challenges before delving into the WSN security requirements, attacks and defenses. The last part of the chapter covers WSN reliability including fault awareness and fault detection in WSNs.

4.1 Why WSNs are Predisposed to Attacks?

In this section, we describe unique properties that predispose WSNs to various types of attacks.

Resource Limitations Many security mechanisms employed in computer networks rely on some form of cryptography. While public key cryptography is in many cases a more versatile security scheme relative to private key cryptography [52], the limited memory and processing power of the WSN nodes mean that the former method can only be used subject to very careful optimization of the algorithms at both the design and implementation levels [64]. This challenge does not arise with traditional computer networks.

Large-scale Deployment Sensor networks typically contain very many nodes (e.g., thousands of nodes) that work in collaboration to achieve the purpose of the network. This means that the two-party functionality seen with many security protocols may not always be suited for WSNs [52]. Further, the large number of nodes sharing sensitive information requires that security designs have to be cognizant of the fact that a single compromised node could leak sensitive information about the entire network.

Open Deployment Because sensors are typically deployed in outdoor unattended environments, an attacker could very easily have physical access to them and extract sensitive information (e.g., security keys) from the motes. This is in contrast to traditional computer networks where it is for the most part safe to assume threat models in which an adversary will have very low likelihood of physically accessing a node.

Wireless Connectivity Because WSNs communicate via wireless communication channels, the adversary only needs to tune into the frequencies being used for communication to be able to eavesdrop on traffic being exchanged between the nodes. Different from a wired computer network, therefore, security mechanisms for WSNs need to take these kinds of attacks into consideration.

The unique attributes such as those described above make possible a number of attacks against WSNs that call for defenses, which are for the most part specifically tailored to WSNs.

4.2 Security Requirements

The fundamental security requirements of a WSN are given below.

Authentication Authentication enables a node to confirm that the identity of another node with which it communicates is as claimed. This helps a node to verify the origin of packets sent to it, ruling out the chance that spoofed packets or packets maliciously injected into the wireless communication channel may be mistaken for genuine packets. A message authentication code (MAC) attached to a message can be used to verify the origin of a message [64].

Confidentiality Confidentiality stipulates that information is only accessed by nodes which are supposed to access it. Confidentiality is ensured by encrypting the packets being sent out at the originating node so that they are decrypted at the receiving node. Depending on the application, encryption could be applied to the data part of a packet or the full packet (including the header) [64]. Encryption of the full packet helps obfuscate the node identities (which are located in the header) which helps minimize the chances that a given node's identity could be spoofed by an eavesdropper.

Integrity Integrity ensures that a message sent between two nodes is not modified by an adversary. If for instance routing or clock synchronization packets exchanged between nodes are modified by a malicious entity, the entire network could come to a halt. Data aggregation (i.e., the combination of readings obtained by multiple sensors) is another WSN service that heavily relies on the assumption that data being exchanged between WSN nodes is not modified maliciously [52]. A keyed checksum (e.g., a MAC) can be used to determine whether messages have been modified [64]. Figure 4.1 illustrates how a MAC appended to a message at the sending node can be used by the receiver to determine the integrity of the message. Taking the message and a key as input, the sender uses a MAC algorithm to compute the MAC, which is then appended to the message before it is sent out to the receiving node. At the receiving node, the message and the key (same as the key which was used at the sending node) are input to the MAC algorithm so as to compute a MAC. If this MAC matches with the one retrieved from the message, then the message is confirmed not to have been tempered with.

Availability Availability indicates that the WSN must be functional at all times and provide services whenever needed. One way in which availability can be achieved is by ensuring reliability at both the node and network level, which can in turn be achieved by building fault tolerance into the individual nodes and the network [27, 29, 57]. Designing the system against different kinds of denial of service attacks is also crucial for the guaranteed availability of the WSN.

Data Freshness WSN systems typically either continuously sense and forward data from the environment, or forward data in response to a certain event. In both cases, it is crucial that data sensed by the nodes reaches the base station as soon as possible (i.e., when is still fresh). This does not only minimize the likelihood that replayed packets sent by an attacker could be mistaken for legitimate packets, but also ensures that the right kinds of interventions can be undertaken to react to the events sensed by the network. In a target tracking application for instance, the target can only be tracked if current data is forwarded to the base station. Data freshness is in general achieved through reliable transport and routing schemes that minimizes packet losses and delays.

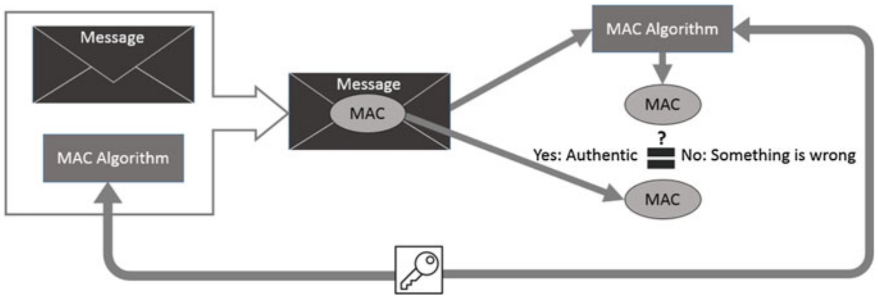

Fig. 4.1 Using a MAC to verify message integrity

4.3 WSN Attacks and Defenses

There are several approaches to the categorization of attacks on WSNs. One of these approaches categorizes attacks based on whether they disrupt the functionality of the network or not. Attacks which disrupt network functionality are called active attacks (e.g., network jamming attacks [59]), while those which do not disrupt network functionality are referred to as passive attacks (e.g., packet eavesdropping attacks [61]). Another common way to categorize WSN attacks is to classify them based on whether they are internal (i.e., launched by nodes which are part of a WSN) or external (i.e., launched by nodes or devices that are not part of the network). The Sybil attack [36] is an example of an internal attack, while a network jamming attack [59] is an example of an attack which would typically be launched by an external agent (i.e., an external attack). It is noteworthy that an attack launched by an external entity which gets authorization to access the network and then exploits the privileges to launch attacks would be classified as an internal attack [64].

The most prominently used approach for the categorization of WSN attacks in the literature distinguishes between attacks based on the layer of the communication architecture which the attack targets. We adopt this approach in this work, since it gives a fine-grained view of the aspects of the different algorithms and (or) protocols that each attack seeks to leverage. For each layer of the network model, we first discuss the potential attacks before delving into the different defenses that could be used to thwart the attacks.

4.3.1 Physical Layer Attacks

We discuss here several physical layer attacks, including tempering, jamming, and eavesdropping and traffic analysis.

Tempering Given physical access to a WSN node, the attacker could temper with the node in several ways, which include reprogramming the node with the aid of easily accessible tools (e.g., UISP [23]), compromising data stored on the nodes (e.g., encryption keys) and the complete physical destruction of a node, to mention but a few [23].

Jamming In this type of attack, the adversary sends out signals (e.g., using a specialized waveform generator [59]) that interfere with the radio frequencies being used by the WSN. Depending on the specific mechanism used to launch the jamming attack, a sizeable portion of the network could be disrupted, rendering nodes unable to send or receive data along the channel [59]. Figure 4.2 illustrates a typical jamming scenario where a jammer interferes with communications associated with all nodes within a certain radius, r, of the jammer.

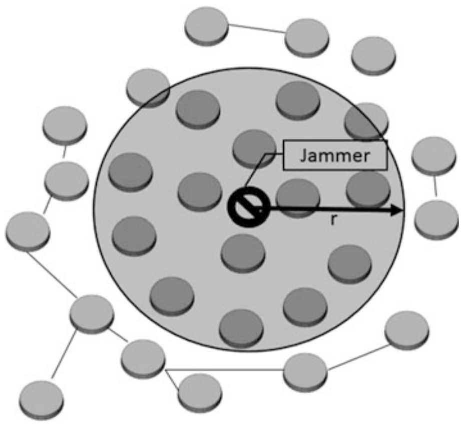

Fig. 4.2 A jamming attack disrupting all communications between nodes within a radius *r* of the jamming node

Eavesdropping and Traffic Analysis Since WSN signals are broadcast in the air, an adversary within the range of the signals could listen to the transactions going on in the wireless channel with the aid of antennas that cost as little as $20 [10]. From data captured during eavesdropping, the adversary could directly extract the message content, or carry out traffic analysis to make inferences such as the location of the base station, which could in turn inform more targeted attacks.

4.3.2 Physical Layer Defenses

One potential defense against tempering is the design of temper-proof WSN nodes (e.g., by encasing them in a physically sturdy package [62]). The challenge with this option, however, is that the cost of each mote could tremendously increase, making the deployment of large WSNs very expensive. Under the assumption that an adversary accessing a WSN node will have to move it (e.g., from one place to another), building location awareness into the applications running on the motes could be used to defend against some physical attacks. If a node is determined to have been moved (e.g., based on accelerometer or GPS sensor readings), its data could for instance be flagged by other nodes as being potentially compromised. Alternatively, the node could be configured to delete sensitive information from memory if movement is detected [62].

Jamming and eavesdropping attacks could be mitigated through the use of spread spectrum communication. The philosophy behind spread spectrum communication is to distribute the communication channel over a large number of frequencies, making it very expensive for the adversary to jam or eavesdrop on all the frequencies. In one spread spectrum technique (called frequency-hopping [66]), senders rapidly switch between frequencies using a pattern known to the receivers. This way, an adversary would not know which frequency to jam or eavesdrop while the receiver would be aware of the variations in transmission frequencies. The challenge with spread spectrum communication in broadcast systems such as WSNs is that the adversary only needs to compromise one node to determine the range of frequencies used by the WSN. Also, spread spectrum functionality increases the power requirements of the nodes, and for a large WSN, can dramatically increase the cost of deploying the network.

The ultimate solution to eavesdropping/traffic analysis on the communication channel and tempering attacks aimed to infer information stored on a captured node is encryption/cryptography. Encryption may be applied to the data stored on the motes, or to the packets being exchanged between nodes over the channel. Because cryptography is a solution to a wide range of other attacks, we will discuss it separately and address its dynamics and the different factors affecting its applicability in WSNs (see Sect. 4.4).

4.3.3 Link Layer Attacks

MAC Protocol Violations MAC protocols generally help ensure that the sensors in the network efficiently use the shared communication channel. When a given node violates the MAC protocol mechanisms (e.g., by sending data during a time slot when another node is supposed to be sending), packet collisions occur. Depending on the extent of the violation, the collisions could result into a wide range of issues, including corruption of the data in the packets, unfair bandwidth usage, and in the worst case, total denial of service if the malicious sender continuously occupies the channel and (or) the attacked nodes continually attempt to retransmit corrupted packets [62].

MAC Identity Spoofing When a node broadcasts data on a WSN, its MAC identity can be accessed by all nodes sharing the communication channel. Given MAC addresses of nodes on a WSN, a malicious node (that may be within range of the network without necessarily being part of the WSN) could use these identities to masquerade as any of these nodes. If, for example, the malicious node masquerades as an aggregation point, it could leverage this role to access privileged resources of the WSN [61]. The Sybil attack [36]—an attack in which a malicious node on the WSN presents different identities to the network at different points in time or cycles through multiple identities to create the impression that they are all simultaneously present on the network—is realized in WSNs through MAC identity spoofing.

4.3.4 Link Layer Defenses

Error correcting codes [51] can be employed to address data corruption issues caused by packet collisions if the extent of collision is low to moderate (e.g., if collisions are comparable to those seen due to probabilistic errors [62]). However, this approach would not handle a heavy volume of collisions as it would require a considerable amount of resources [62]. Incorporating rate-limiting in the MAC protocol to ignore requests beyond a certain threshold of requests and the use of time-division multiplexing to limit the amount of time that nodes can use the channel are the other potential defenses against attacks which overload the channel through violations of the MAC protocol [62].

MAC identity spoofing can be tackled by associating each identity with a secret key, making it impossible for the adversary to use a given address without having its associated key [62]. Under the assumption of an immobile WSN, position information (registered when the network is set up) can be used in conjunction with MAC identities to confirm that a node is not using a spoofed address [36]. Other techniques, such as node registration (at a central server which can be polled to verify the identities of nodes in the network), and code attestation (e.g., to remotely determine the legitimacy of node by comparing its memory contents to those of known legitimate nodes) have also been proposed to address these kinds of attacks [36, 50].

4.3.5 Network Layer Attacks

Routing Manipulation Attacks The routing tables used by nodes to forward packets in a WSN can be poisoned in a number of ways. For example, a malicious node may modify or spoof route update packets before forwarding them to the other nodes in the network. As a result, the nodes receiving these updates may direct traffic along routes determined by the attacker, which could in turn result into congestion and collapse of the network [61]. Some specific types of route-poisoning attacks include:

(a) *Sinkhole* attack: In this attack, the adversary manipulates routing information to lure a large number of nodes into routing their traffic via a node controlled by the adversary [26, 62] (we also alternately refer to this node as the malicious or attacking node).
(b) *Wormhole* attack: This attack is centered on route manipulations designed to make two distant malicious nodes appear to the other nodes to be much closer to each other than is actually the case. Figure 4.3 illustrates the concept of a wormhole attack. The compromised nodes 11 and 2 have a direct (wormhole) path which is much shorter (in terms of number of hops) than other potential paths between the two parts of the network. Nodes 12 and 1 are very likely to communicate via the wormhole path since this path creates the illusion that these notes are very close to each other (in fact, much closer to each other than they actually are). Without the wormhole attack, these two nodes would ordinarily have communicated via the much longer route via nodes 11, 8, 6, 7, 5, 4 and 2.

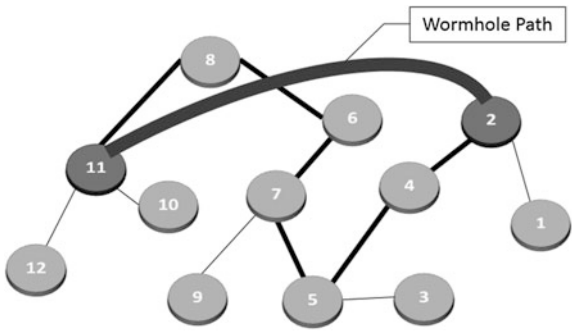

Fig. 4.3 A wormhole attack in a WSN

If the adversary strategically locates the wormhole in such a way to fool the other nodes that they are just a few hops away from the base station via the wormhole, another malicious node placed between the wormhole and the base station can then manifest as a sinkhole [26]. Besides the sheer destabilization of the routing process, attacks such as the sinkhole and wormhole attacks can also be used as a vehicle for eavesdropping given the high amount of traffic traversing the nodes being controlled by the adversary.

(c) *HELLO Flood* attack: This attack basically involves the attacker sending HELLO packets to various nodes in the network to dupe them into classifying the attacking node as their neighbor [26, 62]. Because some of the nodes may, in fact, be very far away from the malicious node, the adversary may have to use a high-powered transmitter (e.g., laptop class device) to achieve the range of transmission required to deliver the HELLO packets. If this malicious node (now perceived to be a neighboring node) broadcasts a low cost route to the base station, WSN nodes may attempt to send data via this route, potentially resulting into failed data transfers, retransmissions and channel congestion (since the offending node is actually not in radio range with many of the nodes attempting to send data via it.). Attacks such as the sinkhole and wormhole attacks could be realized with the aid of a HELLO Flood attack if the malicious node(s) is (are) located in such a way to enable successful data transfers from the victim nodes to the malicious node(s).

(d) *Acknowledgement Spoofing*: When a node *A* sends out packets to node *B*, routing algorithms used by WSNs require that *B* (explicitly or implicitly) sends some form of acknowledgement to *A* if data from *A* is indeed received by *B* [26]. A malicious node *C* that is aware of data being sent from *A* towards *B* (awareness of the transaction arises out of the broadcast nature of the channel) can spoof the identity of node *C* and send acknowledgement information towards *A*, which would then believe that the acknowledgement indeed originated from *B*. This kind of attack could, for instance, fool node *A* to believe that node *B* is alive yet it is in actual sense dead. With node *A* having a wrong view of the network topology, an unstable routing process could result.

(e) *Selective Forwarding*: Rather than forward all received messages as is supposed to be the case in multi-hop networks, a malicious node launching this attack may only forward a subset of the messages [26, 61]. This kind of attack could, for instance, be launched by an adversary who is interested in suppressing the propagation of traffic originating from a certain node or subset of nodes. In the extreme case, a malicious node launching this attack could drop all packets received by it, creating what is termed as a black hole [26]. Figure 4.4 illustrates a black hole in the middle of a WSN. The dark colored nodes route their traffic via the black hole, which drops all packets and denies them access to the base station.

A notable aspect of the black hole variant of selective forwarding attacks however is that neighboring nodes may assume the malicious node to have failed, prompting them to use alternative routes that do not go through the malicious node. More subtle forms of this attack may hence be more attractive for an adversary who is keen to avoid triggering such route modifications that may result into the exclusion of the malicious node. It is noteworthy that selective forwarding attacks require that the malicious node is positioned in such a way to be traversed by the traffic to (or from) the node that the attacker seeks to target. To meet this requirement, these types of attacks may leverage aspects of other attacks, such as the sinkhole, wormhole and Sybil attacks to mention but a few.

Routing Table Overflows Route-poisoning can also be induced by overflowing the routing tables in the victim nodes. In particular, by continually sending void routing information to the WSN, an adversary could ensure that nodes in the network have bogus routing information in their routing tables, with little or no room available in the node buffers for correct routing information [61].

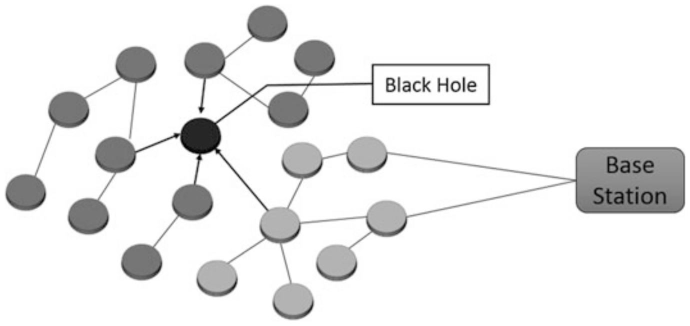

Fig. 4.4 Black hole illustration: all messages received at the black hole are not forwarded to their destinations

4.3.6 Network Layer Defenses

The use of multiple paths for data transfer can limit the effects of some route manipulation attacks (e.g., selective forwarding, and the sinkhole and black hole attacks), since the damage to the system would be restricted to only the traffic traversing a subset of the paths. The challenge with this approach, however, is that the maintenance of potentially redundant routes may have its toll on network resources, let alone the scheme not being feasible in sparse networks which have very few possible routes [46]. Authentication schemes that only update routing table entries after the node originating an update is verified are another defense against these kinds of attacks [61]. At the message level, a message authentication code (MAC) attached to each message (including routing-specific messages) can be used to determine if routing information could have been altered [62]. Against wormhole attacks, precise clock synchronization between communicating nodes could be leveraged to estimate the distance traveled by each packet and rule out the possibility that a given malicious node is much further than it claims to be [61]. Geographic routing protocols that use knowledge of each node's location information as additional information to determine the source of a given packets can help overcome attacks such as the HELLO floods or acknowledgement spoofing [46].

4.3.7 Transport Layer Attacks

Denial of Service As part of the mechanisms to ensure end-to-end reliability, transport layer protocols in WSNs (see survey in [11]) maintain state information (e.g., information on the status and identity of each active connection). When an adversary opens up a large number of connections in a short time, the amount of information stored about the connections at the concerned node(s) increases tremendously, and in the worst case can deplete all memory at the nodes [62].

Connection Desynchronization In this attack, an adversary sends spoofed messages to one or both nodes involved in a connection, fooling them into requesting for retransmission of missed frames. The attacker manages to force the nodes into this synchronization recovery phase by carefully selecting the sequence numbers and control flags of the spoofed packets. The main effect of this attack is the wastage of energy and memory of the nodes during the continued retransmission requests [46].

4.3.8 Transport Layer Defenses

A possible defense against *connection desynchronization* attacks is the authentication of all packets being exchanged during a connection. This helps ensure that packets from a malicious sender cannot be confused with those being sent between the genuine nodes. For the DoS attacks against the transport layer, one proposed solution is to force each node to commit to every connection it initiates, e.g., by having it first solves a puzzle [62]. The puzzles would have the undesired impact of consuming a node's resources, but would make it computationally challenging for a node to initiate a very large number of connections and leave them unclosed [62].

4.3.9 Application Layer Attacks

Attacks on Data Aggregation Process Data aggregation, a service provided by the application layer, ensures meaningful combination of data from the sensors (e.g., by eliminating erroneous readings) to enable the accurate estimation of the sensed environment [52]. Possible attacks on this service include the malicious modification of data before it is forwarded to the base station and the complete disruption of the service by, for instance, strategically placing a black hole in the WSN. With the base station having wrong information about the sensed environment, the network could then be compromised in several other ways if the base station triggers actions based on the wrong information fed to it [61].

Clock Desynchronization Clock synchronization, another service provided by the application layer, typically has to be carried out with very high accuracy [14, 52]. Attacks on this service (i.e., clock desynchronization attacks) can be launched by sending malicious beacon packets into the network, forcing some nodes to adjust their clocks and get out of sync with the other nodes [61]. For example, in WSN applications in which time series data is used to make velocity estimates (e.g., see [7]), clock synchronization problems would result in wrong velocity computations, defeating the very purpose of the network. The identification of duplicate readings from different nodes detecting the same event is another WSN operation which heavily relies on the sensor clocks being closely synchronized [14, 25].

Selective Forwarding This attack is similar to the one already described under the network layer, except that the attacker or compromised node makes forwarding decisions based on the contents of the packets as opposed to the addresses of the nodes involved [61].

4.3.10 Application Layer Defenses

Attacks on data aggregation and clock synchronization are essentially attacks on the integrity of data being exchanged between nodes, meaning they have to be addressed through authentication as data finds its way through the network. The selecting forwarding attack is addressed through encryption schemes that obfuscate the data [61].

4.4 Cryptography in Sensor Networks

Cryptography is at the heart of many WSN security mechanisms designed to ensure data confidentiality and authenticity. The two categories of cryptography—symmetric (or private) and asymmetric (or public) key cryptography—are both used in several cryptography schemes proposed for WSNs. For each of symmetric and asymmetric key cryptography, this section gives an overview of the design considerations and operational dynamics of the different schemes that have been evaluated or proposed for WSNs. A more detailed treatment of these topics can be found in [62, 64].

4.4.1 *Symmetric Key Cryptography in WSNs*

In a symmetric key system, the sender encrypts a message using a secret key that is also known to the receiver of the message. On receiving the cipher text (i.e., encrypted message), the receiver uses the same key to decrypt the message. A major requirement of symmetric key systems is that the keys remain secret, and be changed frequently to minimize the possibility of success of attacks. Further, for large networks, it is crucial that the distribution of keys between the sensor nodes be done as efficiently as possible because the number of key-pairs can be very large. To meet the above two requirements (i.e., key security and efficient key distribution), there is a wide range of techniques (i.e., key management schemes) whose applicability to WSNs has been widely studied in the literature. We briefly discuss these schemes in the rest of the section.

The simplest approach to key establishment in WSNs would be to use a single key that is known to all nodes on the WSN. If one node has to communicate with another, it only has to encrypt data using the global key, so that the receiver can decrypt the data using the same key. The challenge with this approach is that a single compromised node results into the entire WSN being compromised.

Centralized key distribution (e.g., the scheme used in [20]) presents a more secure alternative to the above scheme as each node only shares its key with the base station. Communication between any two WSN nodes begins with each of them communicating with the base station, which sends the symmetric key required for the two nodes to communicate. A weakness with this scheme is that it could incur a huge communication overhead, e.g., when two neighboring nodes have to first communicate with a distant base station before exchanging data with each other. This scheme also faces a single point of failure vulnerability since a breach of the base station exposes the keys for each pair of nodes on the WSN [64].

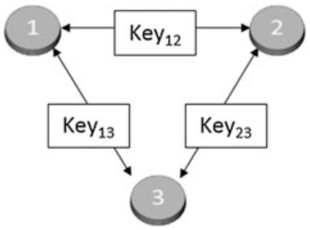

Fig. 4.5 Pairwise key used for communications between each pair of nodes on the network

A naïve approach of solving the single point of failure problems seen with the above two approaches is to pre-distribute the keys, i.e., preload each node pair with a private key before deploying the nodes on a WSN (Fig. 4.5). Assume a WSN having N nodes. Since each node will have to store $N - 1$ keys, a total of $0.5N(N - 1)$ keys will have to be stored in the WSN. As N increases, this number will grow very fast, potentially imposing significant memory constraints on the WSN [64]. To reap the benefits of key pre-distribution while minimizing resource consumption, there is a wide range of key distribution schemes that have been proposed in the literature [3]. We briefly discuss these next.

Random Key Pre-Distribution (RKP) [15] In this approach, each WSN node is preloaded with a key ring, which is a set of keys randomly selected from a large pool of keys. With a certain probability, two nodes will share a key, and hence be able to communicate directly using that key. Figure 4.6 illustrates how RKP works. The nodes numbered 1, 2, 3, 4 and 5 randomly choose keys from a large pool of keys. Node 1's key ring has the keys a, b and c while nodes 2, 3, 4 and 5, respectively, have key rings comprised of the keys p, q and y; a, y and q; p, x and y; and x, y and p. Nodes numbered 1 and 3 have the key a in common, and can hence use it for their pairwise communications. Similarly, nodes 3 and 4 can use the key y for their communications. The other nodes use their common keys for communication in the same way.

It is noteworthy that the absence of direct connectivity between every pair of nodes will not prevent communication between any two nodes because data transfer between a pair of nodes only requires that a path exists between the two nodes via other nodes which share keys pairwise. For example, node 3 can communicate with node 2 through node 5 which shares a key with both nodes. Similarly, node 1 can communicate with node 4 via node 3.

With a WSN of 10,000 nodes, it was shown in [15] that key rings of just 250 keys can guarantee "almost-certain" connectivity for the entire network. This translates into a total of approximately 2.5 million keys, which is a significant reduction in memory requirements if compared to the approximately 50 million keys required (i.e., $0.5N(N - 1)$) for a similarly sized network using the previously described scheme.

Variants of RKP Because the basic RKP scheme requires that two nodes only to have a single shared key to set up a secure communication link, the compromise of a single key ring (i.e., capture or compromise of a single node) exposes a large

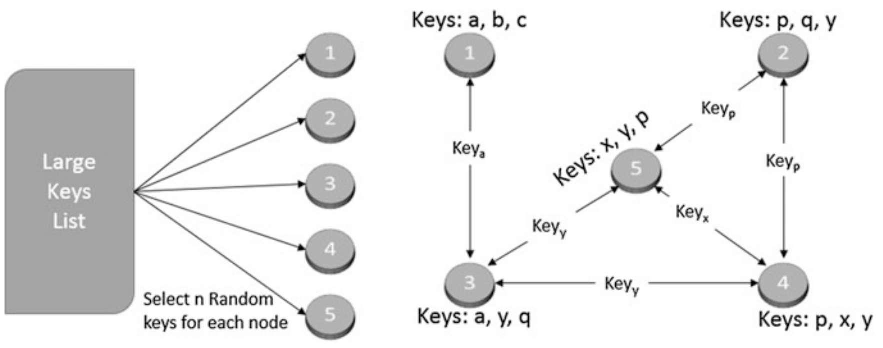

Fig. 4.6 Illustration of random key pre-distribution

number of nodes to attack. The *q*-RKP scheme [8] seeks to improve the resilience of a WSN to attack by requiring that two nodes should share q keys, where $q > 1$ to be able to set up a communication link. As q increases, the chance that an attacker who compromises a set of keys can break a given link decreases exponentially [8]. An augmentation of the *q*-RKP scheme uses multipath reinforcement to improve security [8]. In particular, after the initial secure links have been set up, this scheme requires that the nodes update the keys with random values using multiple independent links. The use of multiple links helps limit the impact of node capture attacks on the network. The use of multiple independent links helps ensure that an attacker listening to the network would have to listen to multiple links to be able to extract enough information to compromise the network. In a location-dependent version of RKP [1, 23], key rings are constituted based on the locations of the sensors in such a way that neighboring nodes have more keys in common than distant nodes. The advantage of this is that the effects of node compromise are constrained to a local area. Several other derivatives of RKP have been proposed (e.g., see [12, 24, 32, 41]) with the major motivation being the improvement of the security, memory requirements and network connectivity properties of the base key pre-distribution protocol.

Deterministic Key Distribution A shortcoming of the RKP protocols is that there is a possibility of certain nodes being isolated from their neighbors due to a lack of shared keys. This weakness follows from the non-guaranteed connectivity depicted by the underlying random graph model on which the base RKP protocol is based. Deterministic key pre-distribution schemes overcome this challenge by imposing some requirements on the extent of connectivity between nodes. In one deterministic key pre-distribution scheme (see [31]), a strongly regular graph is used as the underlying model, guaranteeing that any two adjacent nodes have connectivity to a certain number of neighbors and every two nonadjacent nodes have connectivity to a certain number of neighbors. In another approach to deterministic key pre-distribution (e.g., see [63]), the WSN is modeled using a multidimensional grid in which nodes along the same dimension are able to establish shared keys while nodes with dimension mismatches can negotiate keys through indirect connections.

4.4.2 Asymmetric Key Cryptography in WSNs

To ensure message confidentiality through asymmetric key cryptography, a sender uses the receiver's public key to encrypt a message while the receiver uses their private key to decrypt the message. The private key is only known to the receiver, while the public key can be accessed by any entity which wishes to send a message to the receiver. Asymmetric cryptographic schemes are, however, resource-intensive, as they involve complex computations such as the modular exponentiation of large numbers [9]. In a WSN setting, while the decryption process can for some applications be assumed to be done at the base station (which can be assumed to have a significant amount of resources) [20], the encryption step has to be done on the WSN nodes, which are inherently resource-limited. Asymmetric cryptography, however, has a number of advantages that make it appealing for WSN applications. One major advantage of asymmetric key cryptography relative to symmetric cryptography in WSNs is that it is robust against node capture attacks since a captured node would only reveal its private key and no other information. Other advantages of these schemes include their scalability (since they perform the same way regardless of the number of nodes in the network [9]), and their ease of revocation of compromised key-pairs [9].

Early efforts towards asymmetric cryptography in WSNs worked around the resource-intensiveness of the assymetric algorithms via a scheme which simulates an asymmetric system through a delayed disclosure of symmetric keys in which keys are revealed sequentially over time (scheme is presented in [40]). As argued in [20], this delayed disclosure of keys may require nodes to store multiple messages before they can be authenticated, which can, in turn, take its toll on the resource-constrained nodes. Another challenge posed by the emulated asymmetric cryptography scheme in [40] is that it requires frequent transmission of the keys and key management messages, which has implications on both the energy requirements and security of the network.

On basis of these factors, Gaubatz et al. [20] argue that emulated asymmetric cryptographic schemes in WSNs eliminate many of the benefits of symmetric key cryptography and propose the use of customized hardware to support the expensive asymmetric key operations. In particular, they show using customized hardware that two well–known asymmetric cryptographic schemes—the NtruEncrypt and Rabin algorithms—are feasible for WSN systems. Software-based approaches that minimize the computational load on the nodes (e.g., by among other factors selecting a low value of the exponentiation base, e, used in the cryptographic computations) have also been found to enable the implementation of public key cryptography in WSNs. For example, in [58], an RSA-based method that uses $e = 3$ is shown to work well for the UC Berkeley MICA2 motes. A much wider range of cryptographic schemes is studied in [42] with the authors finding that with careful design, asymmetric cryptography is much more feasible for WSNs than earlier thought.

4.5 Faults in WSNs

Reliable performance of WSNs is necessary in safety-critical applications, such as process control, chemical agents monitoring, radiation contamination monitoring and detection, medical applications and others. Here, we present an overview of statistical data analysis and signal processing techniques that are used to detect sensor faults. Sensor network systems have four levels of abstraction as listed below, and failures can occur at each level.

4.5.1 Fault-Aware WSNs

Fault-aware WSNs have ability to detect, isolate, and mitigate faults at each system level including sensors (components), sensor nodes, and network (system) level. At each level at which faults can occur, fault-aware WSNs are required to detect failures, isolate them and mitigate their effects. Such design improves robustness and reduces the need for hardware redundancy. However, the tradeoffs are usually in higher computation and communication costs that are incurred during fault analysis and mitigation.

Component Level Each sensor node has multiple components, such as sensors, actuators, a power supply, a radio and a microcontroller. Sensors and actuators directly interact with the physical environment and are subjected to noise, aging and errors due to manufacturing defects. The depletion of batteries can cause the sensor nodes to crash. A node with failed radio is equivalent to a crashed node even if the other sensing and power subsystems operate well. Various methods have been proposed to mitigate sensor failures including novel fault tolerance schemes [28, 34] and detection of faulty sensors using Bayesian algorithms [13, 30].

Node Level At this level, data processing includes converting the analog sensor data into digital data. Data encoding and decoding techniques and data compression can be implemented to increase data rate. A node failure may occur because of the physical damage caused by random airdrop or exposure to harmful chemicals. Crash faults in sensor networks are identified in [10]. Fault identification algorithms can be implemented on individual sensor nodes. This increases the storage and computational cost at the node level. On the other hand, this method has the advantage of saving the communication cost required to transmit all the sensor data to the base station. Radio is the most power-consuming component of the sensor node. Applications that perform data aggregation or data processing from multiple

Table 4.1 Example of sensor node energy consumption for Telos-B node

Operation	Energy consumption
Transmit one bit (radio)	2950 nJ/m
Receive one bit (radio)	2600 nJ/m
Execute one operation (processor)	4 nJ/operation

nodes require the base station to inform them about the failed sensor nodes. Such applications would need to remove the failed nodes readings from collaborative operations of sensor nodes. Therefore, there is a two-way communication cost involved in centralized fault detection algorithms. Table 4.1 lists the energy consumed by the Telos-B mote radio and processor [53].

Network Level The nodes form a bidirectional communication link with their peers to achieve a network of interacting sensor nodes. The network layer of a WSN is responsible for efficient communication in the transmission medium. Inability to perform efficient routing tasks may constitute to congestion in the network. Lightweight and energy-efficient transport protocols that can support any generic application have been developed in [49, 55].

System Level For proper functioning of any system, accurate coordination of the distributed actions performed by the diverse system components is needed. The system level analysis involves a centralized method of implementation in which data from all the nodes are gathered at the base station prior to analysis. Algorithms are implemented at a local computer or the data might be transported to remote users via wide area network for analysis and processing. The advantage of this implementation method is that local or remote computers are equipped with more computational and storage resources than a tiny sensor node.

4.5.2 Sensor Faults in WSNs

WSNs can range from a small handful to hundreds or even thousands of sensor nodes with various sensors monitoring diverse physical phenomena. Faulty sensor readings can lead to unnecessary shutdowns of plants or disruptions in monitored processes. While sensor technology advancements can reduce fault occurrences, they cannot be completely eliminated. Here, we present fault detection techniques, lists the common faults for sensors in WSNs, and identify mathematical models for such faults.

Sensor and actuator faults in complex, distributed electromechanical systems are well studied [33, 37, 48]. In [37], a systematic taxonomy of common sensor data faults that occur in deployed sensor networks and the detailed approaches commonly taken to model these faults are provided. These features include characteristics specific to the data, system, or environment pertaining to the system of interest. According to [37], the data features are usually statistical in nature. A confident diagnosis of any single fault may require more than one of these features to be modeled. The fault detection techniques most commonly used include mean and variance (determine expected behavior via regression models or correcting faulty sensor values), correlation (regression methods), gradient (rate of change), and spatial or temporal distance (determines if data is faulty). Table 4.2 shows common fault detection techniques that includes statistical, nearest neighbor, clustering, and classification methods.

Statistical methods create a model of healthy data and then classify any behavior that does not fit that model as a fault. For example, a statistical model of an outdoor temperature sensor may assume that temperature should be high during the day and low at night, and register a fault anytime that trend is not observed. In this case, a fault could either indicate a malfunction in the sensor network or an environmental irregularity, like a storm. Statistical methods have the advantage of justifiable mathematical models, but are not robust to phenomena that do not fit assumed distributions.

Fault classification methods take the opposite approach and instead of creating mathematical models for healthy data, they use mathematical models for faulty data. Incoming data from the WSN is analyzed for behavior that corresponds to the faulty behavior previously modeled. The main advantage of classification methods is that they allow users to specify an exact set of faults to be considered, while the main disadvantage is that they usually cannot detect faults not previously observed.

Nearest neighbor methods forego data models in favor of using a sensor-to-sensor comparison as a fault metric. These methods identify sets of sensors whose data should be similar (i.e., the spatially nearest sensors). Anytime these neighbors report significantly different readings from each other, a fault is registered. The main advantage of nearest neighbor methods is that they allow faults to be detected without a mathematical model for healthy or faulty data. The main disadvantage is that they require appropriate neighbors to be defined, and mistakes in deciding which sensors are related can result in poor data comparisons.

Clustering-based methods compare clusters of nodes rather than sensors. During a learning period, sets of nodes are grouped into clusters, which are then compared to each other to establish which clusters are measuring related data. During fault

Table 4.2 Common fault detection techniques

Method	References	Pros	Cons
Statistical	[3, 60]	Mathematically justified models	Fails if data does not fit assumed distribution
Classification	[35, 44]	Provides exact set of faults	Computationally expensive, requires proper kernel choice
Nearest neighbor	[5, 65]	No assumption on data distribution	Computationally expensive in large networks, dependent on input parameters
Clustering	[45]	No previous knowledge of data statistics is needed, adapts to new data	Cluster width mush be defined

detection, each cluster compares its data with its related clusters, and faults are registered whenever two related clusters are not measuring similar data. Clustering methods are less computationally expensive than nearest neighbor methods. The main disadvantage of clustering methods is the difficulty of appropriately defining clusters.

In [21], Principal Components Analysis (PCA) is used during a learning period to model healthy data behavior with the first four eigenvectors. Using these eigenvectors as a baseline for healthy behavior, any activity that falls outside of the healthy model is considered an outlier. In [35], a Support Vector Machine (SVM) is used to model healthy behavior and detect outliers in real time for a WSN. In [56], possible outliers are sorted and ranked periodically, speeding up the process of outlier discovery for a high dimensional dataset. Calibration methods focus on identifying and accounting for offsets and gains in a WSN. In [6], it is assumed that the phenomena being measured should have similar behavior everywhere and the characteristics of a small base set are used to model the characteristics of the entire network. In [2], it is assumed that the phenomena being measured have some linear correlation between a sensor and its nearest neighbor, and this information is used to calibrate the sensors in the WSN. The NASA Stennis Space Center has implemented a fault detection models for WSNs on a system that consists of multiple wireless sensor nodes, fault detection software, a server, and a client (Fig. 4.7). The system is intended for health monitoring applications of rocket test stands and widely distributed support systems, including pressurized gas lines, propellant delivery systems, and water coolant lines [16–19]. The main system components include a server, a Network Capable Application Processor (NCAP), Communication Module, a set of Transducer Interface Modules (TIMs), and wireless sensor nodes. The TIMs monitor sensors that measure physical phenomena, such as temperature, pressure,

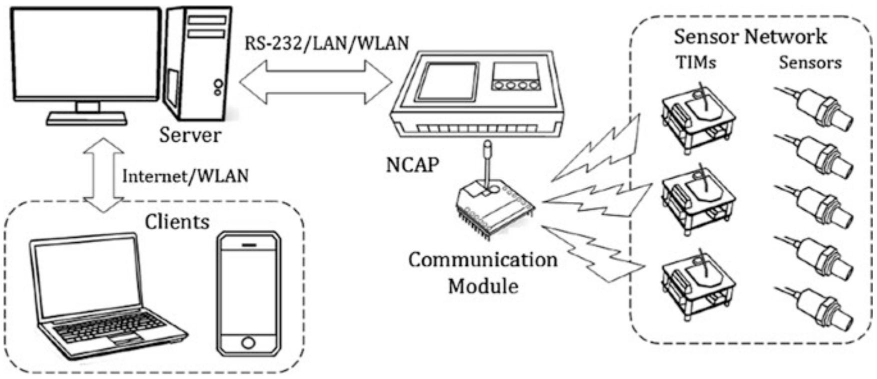

Fig. 4.7 Distributed wireless sensor network configuration that includes a server, a Network Capable Application Processor (NCAP), a communication module, a Transducer Interface Module (TIM) and wireless sensor nodes

and flow rate. They communicate measured data to the NCAP, which analyzes the data for faults. The NCAP records the data (as well as any detected faults) in a server where it can be accessed by a user.

4.5.3 Mathematical Models for Sensor Faults

We discuss here a model that is used in WSN fault detection and identification (Fig. 4.8) [54]. For a sensor node j, sensor readings from the node's multiple sensors are given by

$$Z_j(k) = \left[z_j^1(k)z_j^2(k)\ldots z_j^N(k)\right]^T, \tag{4.1}$$

where individual sensors on a node are identified as $z_j^i(k)$, $i = 1, 2, \ldots, N$. The model considers each node with an equal number of sensors, N, with output values at time instance k given by $z_j^i(k)$ (i-th sensor on j-th node at time k). A true value of a measured physical variable is $u_j^i(k)$ and a faulty measurement of the i-th sensor on the j-th node at time instance k is $\hat{z}_j^i(k)$.

There are two general categories for fault detection techniques: data-centric and system-centric [37]. Data-centric techniques focus on a single data stream to identify faults, while system-centric techniques consider the whole system to detect

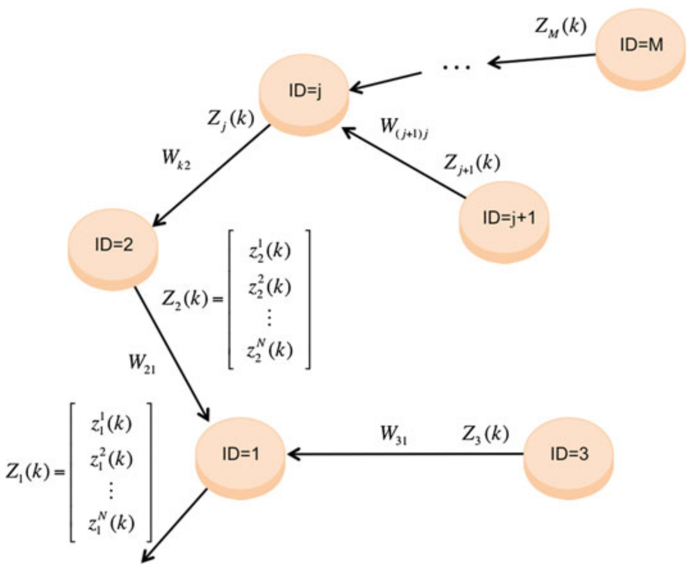
$Z_M(k)$
ID=M
...
ID=j
$Z_j(k)$
$W_{(j+1)j}$
W_{k2}
$Z_{j+1}(k)$
ID=j+1
ID=2
$Z_2(k) = \begin{bmatrix} z_2^1(k) \\ z_2^2(k) \\ \vdots \\ z_2^N(k) \end{bmatrix}$
W_{21}
$Z_1(k) = \begin{bmatrix} z_1^1(k) \\ z_1^2(k) \\ \vdots \\ z_1^N(k) \end{bmatrix}$
ID=1
W_{31}
$Z_3(k)$
ID=3

Fig. 4.8 Model of a WSN that is used in the fault analysis

faults. The two types of techniques need not be used exclusively. A computationally inexpensive data-centric technique may be used to identify abnormal behavior on a sensor stream, at which point a more expensive system-centric technique can be implemented that compares the sensor data to related neighboring nodes, verifying whether the abnormal data reflects an abnormal environment or a faulty sensor reading.

If a model for the behavior of a measured phenomenon is known, the model can be compared to the readings of a sensor, and the error between the two used for identification of faults. If a model is not known, then other techniques listed in Table 4.2 can be applied to identify faults.

4.5.3.1 Data-Centric Fault Models

For each type of fault, there are defining data characteristics, e.g. magnitude for outlier fault or variance for noise fault. One can identify a range for the specific data characteristic under normal operating conditions, and then determine a fault to have occured whenever the characteristic of interest falls outside of the expected range.

The energy constraints in WSNs limit communication and computation complexities, while one of the main hardware constraints is memory. For a large network, it is infeasible to store every data sample collected by a sensor. Therefore, online fault detection algorithms with a sliding data window are preferable ("one-pass algorithms", [22]). A sliding window considers the last W_{sl} data samples from a sensor data stream. Data samples are stored for the duration of the sliding window and then erased. This method is robust to changes in normal behavior of a sensor data.

A less robust and less costly method of defining data norms is to use a learning period. A standard assumption is that sensor data are healthy during the learning period. By analyzing a data stream over the learning period, *normal* behavior can be defined. The advantage is that a normal behavior has to be calculated only once. However, the method is not robust to data streams whose behavior changes with time. For example, the pressure in a pipe may be low during a learning period, but rise when gas is pumped into it. Using a sliding window allows algorithms to update the expected behavior of a data stream as its normal behavior changes, while still detecting true faults in the data stream.

Outlier Fault An outlier is a single data sample whose value is *significantly* (defined by the user) outside of the range defined by previous data samples, see Fig. 4.9. To quantify this range, one can define $\bar{Z}_j^i$, an upper bound of expected sensor data, and $\underline{Z}_j^i$, a lower bound of expected data.

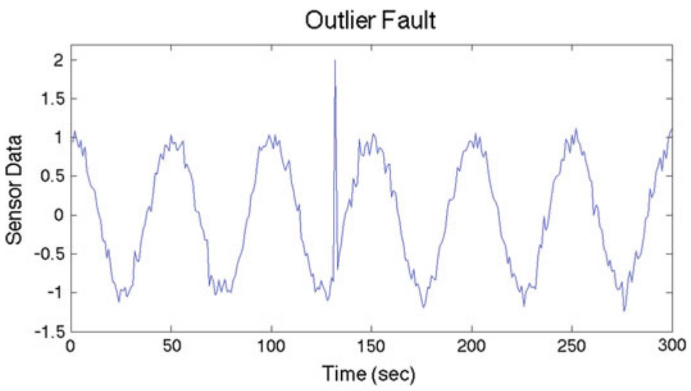

Fig. 4.9 Outlier occurring between 100 and 150 s

The upper and lower bounds $\overline{Z}_j^i$ and $\underline{Z}_j^i$ are given by

$$\overline{Z}_j^i(k) = \max\left\{z_j^i(p)\right\} + c_{\text{out}}\left|\max\left\{z_j^i(p)\right\} - \min\left\{z_j^i(p)\right\}\right|, \tag{4.2}$$

$$\underline{Z}_j^i(k) = \min\left\{z_j^i(p)\right\} - c_{\text{out}}\left|\max\left\{z_j^i(p)\right\} - \min\left\{z_j^i(p)\right\}\right|, \tag{4.3}$$

for $p = k_0, k_0 + 1, \ldots, k_0 + W_{\text{ln}} - 1$ where k_0 is the first data point in the learning period, W_{ln} is the number of data points in the learning period, and c_{out} is a positive constant that determines the outlier detection sensitivity. For instance, $c_{\text{out}} = 0.20$ allows the signal to vary 20 % of the range above and below data limits observed during the learning period. Increasing c_{out} lowers outlier detection sensitivity by allowing some faulty readings to go undetected, while reducing the false alarm rate.

When using a sliding window, $\overline{Z}_j^i(k)$ and $\underline{Z}_j^i(k)$ are recalculated periodically. Bounds can be recalculated every time new data is received. For a sliding window of a length W_{sl}, every time new data is received, the sliding window is updated, and the bounds become functions of the current data sample. The running index p in this case is $p = k - W_{\text{sl}}, k - W_{\text{sl}} + 1, \ldots, k - 1$. The benefit of using such bounds is that if the range of expected data changes, the bounds will adapt to them. However, frequent updating of the signal bounds increases the computational cost and is a natural tradeoff between the false alarm rate and computational cost. The condition for an outlier fault to be triggered is given by

$$z_j^i(k) > \overline{Z}_j^i(k) \quad \text{or} \quad z_j^i(k) < \underline{Z}_j^i(k). \tag{4.4}$$

Spike Fault The term spike fault is used in reference of a small number, r_s, of data points that rise or fall more rapidly than the data during the healthy sensor behavior. Figure 4.10 shows a spike fault where data returns to normal behavior after the spike. The following parameters are used to identify a spike fault: $\overline{R}_j^i$, an upper

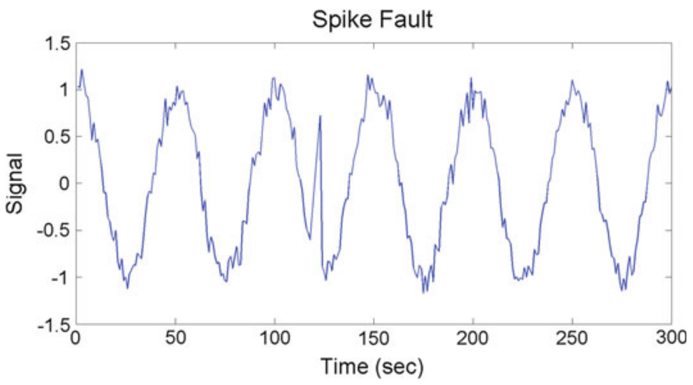

Fig. 4.10 Spike occurring between 100 and 150 s

bound on the gradient which may be ascribed to healthy data on the *i*-th sensor at the *j*-th node, and r_s, a number of successive samples required to have an unhealthy gradient before a spike occurs. The bound that is used in determining healthy behavior is based on a learning period or a sliding window

$$\overline{R}_j^i = (1 + c_{\text{spk}})\max\left|z_j^i(p) - z_j^i(p-1)\right| \tag{4.5}$$

where $p = k_0, k_0 + 1, \ldots, k_0 + W_{\text{ln}} - 1$ in case of a fixed learning period and $p = k - r_{\text{s}} - W_{\text{sl}}, k - W_{\text{sl}} + 1, \ldots, k - r_{\text{s}} - 1$ in case of a sliding window, and c_{spk} is a positive constant that determines the detection sensitivity.

In case of using a sliding window, the gradient bound is a function of sample time *k*. The end of the sliding window should be set more than r_{s} samples away from the current data sample. The number of successive samples, r_{s}, required for a spike fault is influenced both by the phenomena being measured and the sampling rate. Some phenomena, such as the intensity of light, may be expected to produce signals with sharp gradients [37] while others are expected to change smoothly. For a slow-varying signal, a spike could indicate a fault in the sensor or that the sampling rate is too slow. A small r_{s} hastens the detection of spikes, which is crucial in time constrained applications. Conditions for spike detection are given by:

$$\begin{array}{lcl} z_j^i(k) - z_j^i(k-1) > \overline{R}_j^i & & z_j^i(k) - z_j^i(k-1) < -\overline{R}_j^i \\ z_j^i(k-1) - z_j^i(k-2) > \overline{R}_j^i & \text{or} & z_j^i(k-1) - z_j^i(k-2) < -\overline{R}_j^i \\ \vdots & & \vdots \\ z_j^i(k - r_s + 1) - z_j^i(k - r_s) > \overline{R}_j^i(k) & & z_j^i(k - r_s + 1) - z_j^i(k - r_s) < -\overline{R}_j^i(k) \end{array} \tag{4.6}$$

There are two sets of conditions because a spike fault model requires r_s successive points to have either an abnormally large positive or negative gradients. Moreover, this eliminates false positive spike identification when high frequency noise is present, resulting in sensor readings having large gradients in random directions.

Variance Fault Variance fault is defined as a set of data whose values tend to differ from the mean by an abnormally large or small amount, where the threshold parameters are chosen by the user. Defining the length of the window over which variance is calculated as W_v, and the expected value of data over that window as $\bar{z}_j^i$, the variance is then given by [38]:

$$V_j^i(k) = \frac{1}{W_v - 1} \sum_{p=k-W_v}^{k-1} \left(z_j^i(p) - \bar{z}_j^i \right)^2. \tag{4.7}$$

There are two types of variance faults: low variance and high variance faults. A low variance fault (also called a stuck-at fault) occurs when the variance of a data stream is abnormally low, i.e., when a signal is stuck at a certain value. A high variance fault occurs when the variance is abnormally high. The model parameters include $\underline{V}_j^i$ and $\overline{V}_j^i$, a lower and an upper bound on signal variance under healthy conditions.

The variance fault has occurred if $V_j^i(k) > \overline{V}_j^i(k)$ or $V_j^i(k) < \underline{V}_j^i(k)$. Increasing the learning or sliding window size provides a larger sample size for determining the variance. If the phenomenon being measured is slow-varying throughout learning period, it will result in a more accurate representation of the steady state variance of the system. However, if the phenomenon is fast-varying over the learning period, signal changes will be reflected in the healthy variance parameters. A small window size will emphasize the role of variance as imperfections in the system, but will also increase false positives. Figures 4.11 and 4.12 show low variance and high variance faults, respectively.

There are several possible causes of variance faults. The term "noise" is often used to describe some high variance faults in the case of signals with particular

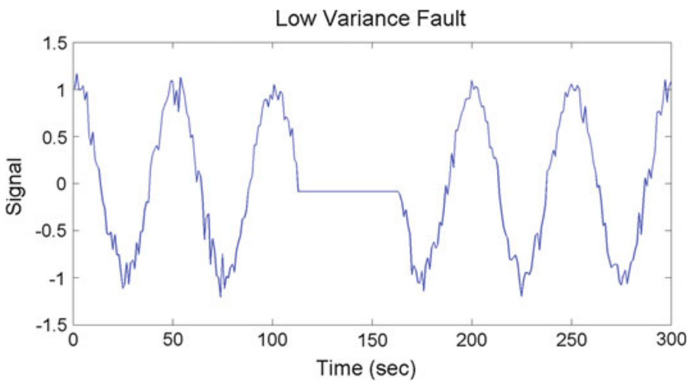

Fig. 4.11 Low variance fault occurring between 120 and 160 s

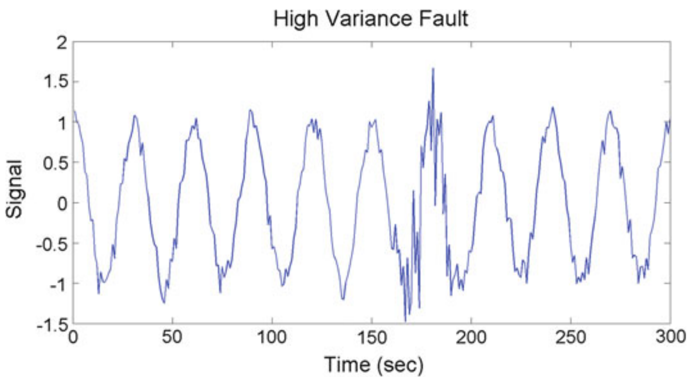

Fig. 4.12 High variance fault occurring between 150 and 200 s

mathematical attributes. Several types of noise exist, such as white noise, pink noise (equal power in bandwidths that are proportionally wide, see below), and violet noise (increases with frequency). Noise is usually classified by its behavior in the power spectrum and the fast Fourier transform (FFT) is often used to detect and isolate a noise fault [43].

High Frequency Noise Fault Noise classification and quantification is usually done in the frequency domain where the FFT is used to obtain the signal power spectrum, Fig. 4.13. Here, we describe a method for detecting undesired high frequency signals in the power spectrum and how to distinguish the presence of unexpected high frequency useful data and high frequency noise.

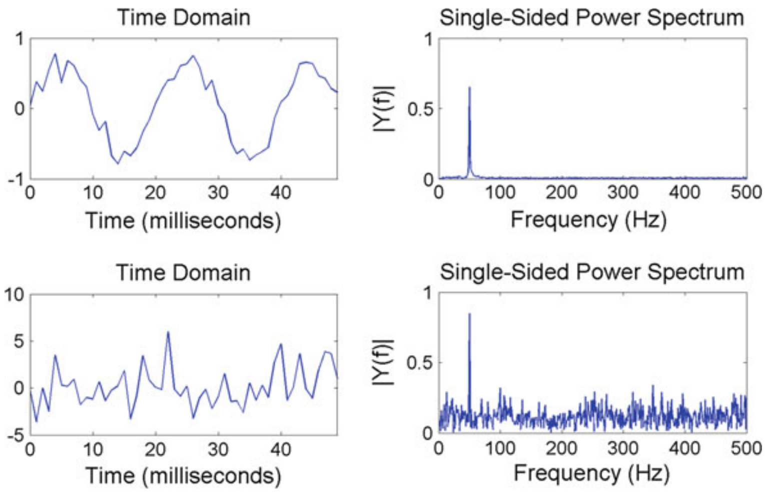

Fig. 4.13 Time domain and power spectrum for a sinusoid with small amount of white noise (*top*) and large amount of white noise (*bottom*)

Figure 4.13 indicates how the FFT can be used to detect noise in a sensor signal. Define the window of data on which the FFT is performed as W_{fft}. The variables that define a high frequency noise fault include $\overline{Y}_j^i$ (the maximum power level classified as noise in the power spectrum) and $\bar{v}_j^i$ (the highest frequency of a useful signal with a power spectrum contribution above the noise level $\overline{Y}_j^i$). To define what constitutes a meaningful contribution to the power spectrum, one needs to compute the power spectrum of a signal over a range of frequencies higher than $v = v_{\max}$. It is assumed that $v_{\max}$ is significantly higher than any frequency that contributes more than noise to the power spectrum. In particular, it is assumed that the highest frequency component is at least W_{pow} higher than any frequency that makes a meaningful contribution to the power spectrum. The maximum power level which noise is expected to contribute, $\overline{Y}_j^i$, is then based on the power spectrum contributions of the last W_{pow} frequency components in the FFT:

$$\overline{Y}_j^i = \max\left\{Y_j^i(v)\right\} + c_{\text{fft}}\left|\max\left\{Y_j^i(v)\right\} - \min\left\{Y_j^i(v)\right\}\right|, v \in (v_{\max} - W_{\text{pow}}, v_{\max}) \tag{4.8}$$

where c_{fft} is the parameter that is used to tune the sensitivity of the high frequency noise model. Any frequency component with a power spectrum contribution at or below $\overline{Y}_j^i$ is considered noise, and any frequency component which contributes a power level above $\overline{Y}_j^i$ is said to have a meaningful contribution to the data stream. The highest contributing frequency component under healthy conditions as the highest frequency v with a power spectrum contribution above $\overline{Y}_j^i$ is defined as:

$$\bar{v}_j^i = \max\left\{v_j^i | Y_j^i(v) > \overline{Y}_j^i\right\}. \tag{4.9}$$

A high frequency noise fault is then defined as the phenomenon wherein the power spectrum of a signal has a meaningful contribution at a frequency higher than the highest frequency expressed during the learning period, $Y_j^i(v) > \overline{Y}_j^i$ for $v > \bar{v}_j^i$.

4.5.3.2 System-Centric Fault Models

System-centric fault models use data from multiple sensors in the system to detect faults. The following techniques require at least one healthy sensor to be correlated to the faulty sensor. This in turn requires that the nodes in the WSN have multiple sensors that measure the same (or related) phenomena, i.e., those nodes are densely deployed to the point of oversampling phenomena of interest [2]. To determine if two or more sensors measure related data, one can use variograms. Variograms quantify the correlation of a phenomenon at one point in the system with a phenomenon at another point in the system. A variogram $\gamma_{j,l}^{i}$ between sensors j and l is defined as:

$$\gamma_{j,l}^{i}(k) = \frac{1}{2W_{\text{vgm}}} \sum_{p=k}^{k-W_{\text{vgm}}+1} \left(z_{j}^{i}(p) - z_{l}^{i}(p)\right)^{2}, \tag{4.10}$$

where W_{vgm} is the window of data where the variogram is being calculated. In the event of spatial correlation among sensors, the variogram is a function of radius r around sensor j:

$$\gamma_{j}^{i}(k) = \frac{1}{2W_{\text{vgm}}\left|\Omega_{j}(r)\right|} \sum_{q\in\Omega_{j}(r)} \sum_{p=k}^{k-W_{\text{vgm}}+1} \left(z_{j}^{i}(p) - z_{q}^{i}(p)\right)^{2}, \tag{4.11}$$

where r is the radius around sensor j, $\Omega_{j}(r)$ is the set of all neighbors of sensor j within radius r, and $|\Omega|$ is the cardinality of the set Ω. A small variogram implies a high correlation among the sensors. If the variogram between a sensor and its neighbors is smaller than some threshold Γ, then the sensor reading is related to its neighbors.

Offset Fault An offset fault occurs when sensor data values are *offset* from the true phenomenon being measured by a constant value:

$$\hat{z}_{j}^{i}(k) = f\left(u_{j}^{i}(k)\right) + \beta_{0}, \tag{4.12}$$

where the function f represents a nonlinear sensor model, $u_{j}^{i}(k)$ is a true value being measured on the i-th sensor at the j-th node, and β_{0} is the offset. Figure 4.14 shows an example of an offset fault. To determine the offset value, one usually requires

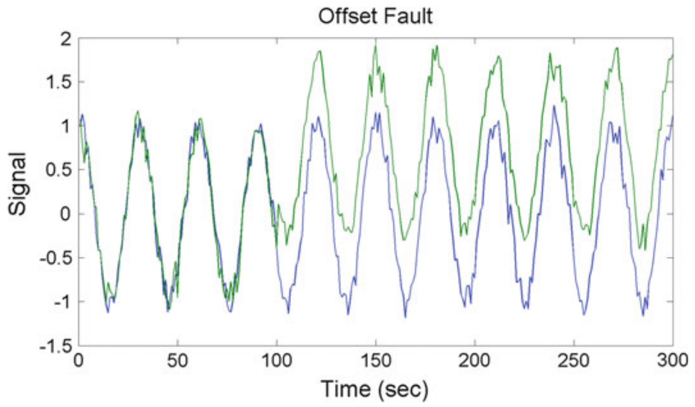

Fig. 4.14 A signal with a constant offset fault

either a ground truth value of the sensor readings or a precise sensor model f, see [16]. In case of a densely distributed sensor network, an offset fault can be modeled as a constant difference between the readings of a sensor and the readings of its related neighbors.

Let us consider the difference between a sensor's readings and the average of the readings of its related neighbors:

$$\Delta_j^i(k) = z_j^i(k) - \frac{1}{|\Omega_j|} \sum_{q \in \Omega_j} z_q^i(k). \tag{4.13}$$

A constant offset is modeled as a slow-varying difference between a sensor's measurements and the true value being measured. To distinguish between a constant offset and a time-changing offset (which may be considered a drift fault, see below) the variance, $\operatorname{var}\left(\Delta_j^i(k)\right)$, over a window of a certain size is observed. The conditions for an offset fault are that the difference between a sensor's readings and the readings of its related neighbors be above the threshold value and that the variance of the difference is sufficiently small, i.e., $\Delta_j^i(k) \geq c_{\text{oft}}$ and $\operatorname{var}\left(\Delta_j^i(k)\right) \leq \beta_{\text{oft}}$, where c_{oft} and β_{oft} are model parameters.

Gain Fault A gain fault occurs when the sensor data for a certain measured phenomenon differ from the healthy sensor data by a constant ratio; see Fig. 4.15. In the event of a gain fault, it is expected that the gain of a sensor's readings compared to the average of its neighbor's readings, η_j^i, to be significantly lower or higher than 1. This ratio can be found as

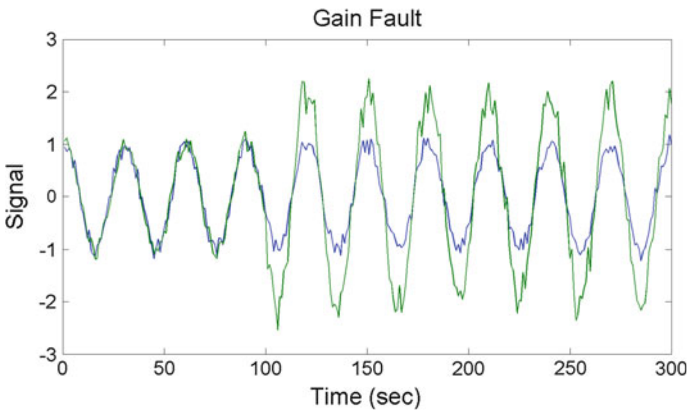

Fig. 4.15 A signal with the gain fault (higher gain in this case)

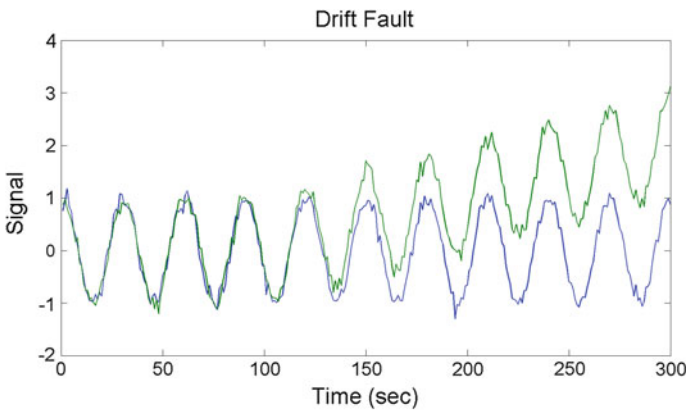

Fig. 4.16 Drift fault when the offset between the signals is steadily increasing

$$\eta_j^i(k) = \frac{z_j^i(k)}{\frac{1}{|\Omega_j|}\sum_{q\in\Omega_j} z_q^i(k)}. \tag{4.14}$$

To quantify variation of the sensor's gain, we consider the variance $\mathrm{var}\left(\eta_j^i(k)\right)$ over the window of data. A gain fault has occurred when $\left|\eta_j^i(k) - 1\right| \geq c_{\mathrm{gn}}$ and $\mathrm{var}\left(\eta_j^i(k)\right) \leq \beta_{\mathrm{gn}}$, where c_{gn} and β_{gn} are small, adjustable parameters.

Drift Fault A drift fault occurs when sensor readings drift away from the real values by an amount that increases with time, Fig. 4.16.

Given $\Delta_j^i(k)$ in (4.13), a drift fault model represents a steady increase in time, i.e., an offset increment $\lambda_j^i(k,L) = \Delta_j^i(k) - \Delta_j^i(k-L)$ is a function of time increment L:

$$\lambda_j^i(k,L) = \lambda_j^i(L) = c_{\mathrm{dft}}. \tag{4.15}$$

If c_{dft} varies with time, then (4.15) models a drift that is increasing according to the integral of c_{dft}.

Questions and Exercises

1. What are the advantages of random key pre-distribution (RKP) relative to centralized key distribution in WSNs?
2. Briefly describe the shortcomings of RKP and the general philosophy of operation of some of the schemes that have been designed to address these shortcomings.

3. Briefly describe the steps through which a keyed checksum can be used to verify the integrity of messages exchanged between WSN nodes.
4. Under what conditions would it be feasible to apply asymmetric key cryptography in WSNs?
5. Use MATLAB to simulate an RKP scheme in which 1000 nodes create key rings containing m keys drawn from a pool of 50 keys for m = 2, 5, 10 and 20. Comment on the connectivity of the network for the different values of m. Assume the keys are the numbers 1 through 50.
6. What is the difference between a wormhole attack and a black hole attack? In your opinion which of these attacks would be more difficult to detect in practice? Give a reason for your answer.
7. Assume that sensor data follow sinusoidal function in time with amplitude of 10. Give an example of an outlier sensor fault and how would you set the threshold parameters in such a case? Does the sampling rate affects the fault model and how?
8. What would be the scenario where the spike fault would overlap with the outlier fault? Can this be prevented?
9. How does the variance fault differ from the high frequency noise fault? Which class set is a superset of the other?
10. Use MATLAB to simulate a high frequency noise fault for a simple data set you create. Then tune fault parameters such that the fault can be effectively detected.
11. Suppose that a sensor is producing sinusoidal type of data over time $z_1(t) = 5\sin(wt)$. At a certain moment there is a gain fault of value 2. How would a faulty data look like and what other faults could be triggered at that moment? Would any other fault be triggered at steady state value of sensor data, i.e., after fault has already occurred?
12. Consider a real life scenario of a sensor with saturation limits at the output. If the drift fault occurs, after some time, what would such fault look like? Any fault detection after long time period will identify what kind of sensor problem?

References

1. F. Anjum, “Location dependent key management using random key-predistribution in sensor networks,” in *Proc. 5th ACM Workshop on Wireless Security*, Los Angeles, California, 2006.
2. L. Balzano and R. Nowak, “Blind calibration of networks of sensors: Theory and algorithms,” in *Networked Sensing Information and Control*, V. Saligrama, Ed. Springer US, 2008, pp. 9–37.
3. L. Bettencourt, A. Hagberg, and L. Larkey, “Separating the wheat from the chaff: Practical anomaly detection schemes in ecological applications of distributed sensor networks,” in *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems*, 2007.
4. R. Beyah, J. McNair, and C. Corbett, Ed., *Security in Ad Hoc and Sensor Networks*, World Scientific Publishing Co, Singapore, 2010.

5. J. Branch, B. Szymanski, C. Giannella, and R. Wolff, “In-network outlier detection in wireless sensor networks,” in *Proceedings of the IEEE Conference on Distributed Computing Systems*, 2006.
6. V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, “A collaborative approach to in-place sensor calibration,” in *Proceedings of the Second International Workshop on Information Processing in Sensor Networks* (IPSN, 2003, pp. 301–316.
7. A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, “Habitat monitoring: application driver for wireless communications technology,” *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 20–41, 2001.
8. H. Chan, A. Perrig, and D. Song, “Random Key Predistribution Schemes for Sensor Networks,” in *Proc. 2003 IEEE Symposium on Security and Privacy*, 2003.
9. H. Chan, A. Perrig, and D. Song, “Key Distribution Techniques for Sensor Networks,” *Wireless Sensor Networks*, C. S. Raghavendra, K. Sivalingam, and T. Znati, Eds., ed: Springer US, 2004, pp. 277–303.
10. S. Chessa and P. Santi, “Crash Fault Identification in Wireless Sensor Networks,” *Computer Communications*, vol. 25, no. 14, pp. 1273–1282, 2002.
11. W. Chonggang, K. Sohraby, L. Bo, M. Daneshmand, and H. Yueming, “A survey of transport protocols for wireless sensor networks,” IEEE Network, vol. 20, pp. 34–40, 2006.
12. W. Du, J. Deng, Y.S. Han, and P.K. Varshney, “A pairwise key pre-distribution scheme for wireless sensor networks,” in *Proc. the 10th ACM Conference on Computer and Communications security*, Washington D.C., USA, 2003.
13. E. Elnahrawy and B. Nath, “Clearing and Querying Noisy Sensors,” *Proc. Workshop on Sensor Networks and Applications (WSNA)*, 2003.
14. J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, 2002.
15. L. Eschenauer and V.D. Gligor, “A key-management scheme for distributed sensor networks,” in *Proc. 9th ACM Conference on Computer and Communications Security*, Washington, DC, USA, 2002.
16. F. Figueroa, J. Schmalzel, J. Morris, M. Turowski, and R. Franzl, “Integrated system health management: Pilot operational implementation in a rocket engine test stand,” in *AIAA Infotech@Aerospace 2010*, Atlanta, GA, April 2010.
17. F. Figueroa, J. Schmalzel, R. Aguilar, M. Shwabacher, and J. Morris, “Integrated system health management (ISHM) for test stand and j-2x engine: Core implementation,” in *44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Hartford, CT, July 2008.
18. F. Figueroa, J. Schmalzel, M. Walker, M. Venkatesh, R. Kapadia, J. Morris, M. Turowski, and H. Smith, “Integrated system health management: Foundational concepts, approach, and implementation,” *NASA Stennis Space Center, Tech. Rep.*, April 2009.
19. F. Figueroa, J. Schmalzel, R. Aguilar, M. Shwabacher, and J. Morris, “Tutorial integrated systems health management (ISHM),” in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011)*, San Diego, CA, June 2011.
20. G. Gaubatz, J.-P. Kaps, and B. Sunar, “Public Key Cryptography in Sensor Networks—Revisited,” *Security in Ad-hoc and Sensor Networks*, vol. 3313, C. Castelluccia, H. Hartenstein, C. Paar, and D. Westhoff, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 2–18.
21. J. Gupchup, A. Terzis, R.C. Burns, and A.S. Szalay, “Model-based event detection in wireless sensor networks,” *Computing Research Repository* arXiv:0901.3923, 2009.
22. C. Han, L. Xu, and G. He, “Mining recent frequent itemsets in sliding windows over data streams,” *Computing and Informatics*, vol. 27, pp. 315–339, 2008.
23. C. Hartung, J. Balasalle, and R. Han, “Node Compromise in Sensor Networks: The Need for Secure Systems,” University of Colorado at Boulder Technical Report CU-CS-990-052005.
24. J. Hwang and Y. Kim, “Revisiting random key pre-distribution schemes for wireless sensor networks,” in *Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, Washington DC, USA, 2004.

25. C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed diffusion: a scalable and robust communication paradigm for sensor networks,” in *Proc. 6th Annual International Conference on Mobile Computing and Networking*, Boston, Massachusetts, USA, 2000.
26. C. Karlof and D. Wagner, “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures,” in *Proc. First IEEE International Workshop on Sensor Network Protocols and Applications*, 2002, pp. 113–127.
27. F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, “Fault tolerance techniques for wireless ad hoc sensor networks,” *Proceedings of IEEE Sensors*, 2002, pp. 1491–1496 vol. 2.
28. F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, “Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks,” *IEEE Sensors*, vol. 2, pp. 1491–1496, 2002.
29. B. Krishnamachari and S. Iyengar, “Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks,” *IEEE Transactions on Computers*, vol. 53, pp. 241–250, 2004.
30. B. Krishnamachari and S. Iyengar, “Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks,” *IEEE Transactions on Computers,* vol. 53, no. 3, 2004.
31. J. Lee and D. Stinson, “Deterministic Key Predistribution Schemes for Distributed Sensor Networks,” *Selected Areas in Cryptography*, vol. 3357, H. Handschuh and M. A. Hasan, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 294–307.
32. D. Liu and P. Ning, “Establishing pairwise keys in distributed sensor networks,” in *Proc. 10th ACM Conference on Computer and Communications Security*, Washington D.C., USA, 2003.
33. X. Luo, M. Dong, and Y. Huang, “On distributed fault-tolerant detection in wireless sensor networks,” *IEEE Transaction on Computers*, vol. 55, no. 1, pp. 58–70, 2006.
34. K. Marzullo, “Tolerating Failures of Continuous Valued Sensors,” *ACM Transactions on Computer Systems*, vol. 8, pp. 284–304, 1990.
35. M.S. Mohamed and T. Kavitha, “Outlier detection using support vector machine in wireless sensor network real time data,” *IEEE Journal of Soft Computing and Engineering*, vol. 1, no. 2, pp. 68–72, May 2011.
36. J. Newsome, E. Shi, D. Song, and A. Perrig, “The Sybil attack in sensor networks: analysis and defenses,” *Proc. Third International Symposium on Information Processing in Sensor Networks*, 2004, pp. 259–268.
37. K. Ni, N. Ramanathan, M.N.H. Chehade, L. Balzano, S. Nair, S. Zahedi, G. Pottie, M. Hansen, M. Srivastava, and E. Kohler, “Sensor network data fault types,” *ACM*, vol. 5, no. 3, pp. 1–29, August 2009.
38. R.L. Ott and M.T. Longnecker, *An Introduction to Statistical Methods and Data Analysis*, Cengage Learning, 6th Edition, 2008.
39. A.S.K. Pathan, H.-W. Lee, and C.S. Hong, “Security in wireless sensor networks: Issues and challenges,” *Proc. 8th International Conference on Advanced Communication Technology, ICACT* 2006, 2006.
40. A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D. Culler, “SPINS: Security Protocols for Sensor Networks,” *Wireless Networks*, vol. 8, pp. 521–534, 2002.
41. R.D. Pietro, L.V. Mancini, and A. Mei, “Random key-assignment for secure Wireless Sensor Networks,” in *Proc. 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, Fairfax, Virginia, 2003.
42. K. Piotrowski, P. Langendoerfer, and S. Peter, “How public key cryptography influences wireless sensor node lifetime,” in *Proc. 4th ACM Workshop on Security of Ad Hoc and Sensor Networks*, Alexandria, Virginia, USA, 2006.
43. J. Proakis and D. Manolakis, *Digital Signal Processing*, Prentice Hall, 2006.
44. S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, “Quarter sphere based distributed anomaly detection in wireless sensor networks,” in *Proceedings of the IEEE International Conference on Communications*, 2007.
45. S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, “Distributed anomaly detection in wireless sensor networks,” in *Proceedings of the IEEE International Conference on Communication Systems*, 2006.

46. D.R. Raymond and S.F. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses," *IEEE Pervasive Computing*, vol. 7, pp. 74–81, 2008.
47. "Coremicro reconfigurable embedded smart sensor node (CRE-SSN) product brochure, url: http://americangnc.com/images/cre-ssn brochure.pdf, accessed Feb. 2015." American GNC Corporation, Tech. Rep., 2012.
48. M. Russell, G. Lecakes, S. Mandayam, and S. Jensen, "The intelligent valve: A diagnostic framework for integrated system-health management of a rocket-engine test stand," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 4, pp. 1489–1497, April 2011.
49. Y. Sankarasubramaniam, Ö.B. Akan, and I.F. Akyildiz, "ERST: Event-to-Sink Reliable Transport in Wireless Networks," *ACM MobiHoc'03*, 2003.
50. M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote Software-Based Attestation for Wireless Sensors," in *Security and Privacy in Ad-hoc and Sensor Networks*, vol. 3813, R. Molva, G. Tsudik, and D. Westhoff, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 27–41.
51. B. Shen, "Application of Error Correction Codes in Wireless Sensor Networks," Master of Science (MS), The University of Maine, 2007.
52. E. Shi and A. Perrig, "Designing secure sensor networks," *Wireless Communications, IEEE*, vol. 11, pp. 38–43, 2004.
53. M. Srivastava, "Energy Aware Wireless Sensor and Actuator Networks," CENS, 2005.
54. N. Vosburg, R. Selmic, S. Oonk, and F. Maldonado, "Intelligent distributed and ubiquitous health management system: Data storage and processing," in *AIAA Infotech@Aerospace*, Boston, MA, August 2013.
55. C.Y. Wan, A.T. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, 2002.
56. Y. Wang, S. Parthasarathy, and S. Tatikonda, "Locality sensitive outlier detection: A ranking driven approach," in *Proc. IEEE 27th International Conference on Data Engineering (ICDE)*, 2011, pp. 410–421.
57. T.-Y. Wang, L.-Y. Chang, D.-R. Duh, and J.-Y. Wu, "Fault-tolerant decision fusion via collaborative sensor fault detection in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 2, pp. 756–768, 2008.
58. R. Watro, D. Kong, S.-F. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," in *Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, Washington DC, USA, 2004.
59. X. Wenyuan, M. Ke, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE Network*, vol. 20, pp. 41–47, 2006.
60. W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng, "Outlying and boundary data detection in sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1145–1157, 2011.
61. K. Xing, S. Srinivasan, M.M. Rivera, J. Li, and X. Cheng, "Attacks and Countermeasures in Sensor Networks: A Survey," in Network Security, S. C. H. Huang, D. MacCallum, and D.-Z. Du, Eds., ed: Springer US, 2010, pp. 251–272.
62. W. Yong, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *Communications Surveys & Tutorials, IEEE*, vol. 8, pp. 2–23, 2006.
63. Z. Yun and F. Yuguang, "Scalable and deterministic key agreement for large scale networks," *IEEE Transactions on Wireless Communications*, vol. 6, pp. 4366–4373, 2007.
64. Z. Yun, F. Yuguang, and Z. Yanchao, "Securing wireless sensor networks: a survey," *Communications Surveys & Tutorials, IEEE*, vol. 10, pp. 6–28, 2008.
65. K. Zhang, S. Shi, H. Gao, and J. Li, "Unsupervised outlier detection in sensor networks using aggregation tree," in *Proceedings of the Advanced Data Mining and Applications*, 2007.
66. S. Zoican, "Frequency hopping spread spectrum technique for wireless communication systems," in *Proc. 5th International Symposium on Spread Spectrum Techniques and Applications*, pp. 338–341 vol. 1, 1998.