Z. J. Haas, L. Yang, M-L. Liu, Q. Li, and F. Li, "Current Challenges and Approaches in Securing Communications for Sensors and Actuators," Chapter 17 in "The Art of Wireless Sensor Networks," H.M. Ammari (ed.), Springer-Verlag Berlin Heidelberg, 2014, DOI: 10.1007/978-3-642-40009-7_17

"Current Challenges and Approaches in Securing Communications for Sensors and Actuators"

Zygmunt J. Haas, Lin Yang, Meng-Ling Liu, Qiao Li, and Fangxin Li

Wireless Networks Laboratory (WNL)
Cornell University, Ithaca, NY, 14853
wnl.ece.cornell.edu

Abstract

Recent advances in MEMS hardware have enabled small-footprint and inexpensive sensors to be deployed in hard-to-access locations and to form wireless sensor networks (WSNs). WSNs are typically mission-oriented networks and offer appealing solutions to a range of practical problems. However, due to the characteristics of WSN, their design principles differ from other types of networks. For instance, the severe limitations of computational and energy resources in the network nodes restrict their ability to process and communicate information. These characteristics, particular to WSNs, dictate new security challenges and require new approaches to implementation of security protocols. In this chapter, we present some of the WSNs security challenges and discuss a number of selected solutions presented in the technical literature.

The structure of the chapter is as follows. In Section 1, we provide background material on WSN security; in particular, we present the security goals, implementation constraints, potential attacks and defenses, and evaluation benchmarks. In Section 2, we discuss basic security challenges and approaches, including cryptography schemes, key management schemes, and attack detection and prevention mechanisms. Then, in Sections 3, 4, and 5, we discuss secure routing, secure localization, and secure data aggregation, respectively. Finally, we conclude the survey in Section 6.

1. Background

1.1 Security Goals:

The field of Information Security defines numerous goals with respect to protecting information, with the following being considered the top three: *confidentiality*, *integrity*, and *availability* [1]. (These three components, being the core principles of information security, are often referred to as the *CIA triad* in the *Information Assurance* field [2].) Information confidentiality is defined as the concealment of information, so that the information can only be available and disclosed to the intended parties. Integrity of

information refers to its trustworthiness, so that an adversary cannot create, modify, or destroy information, including protection against injection of fraudulent, duplicate, or old (expired) information (e.g., the *replay* attack). Information availability denotes the ability of timely and reliable access to the information. The operation of averting information availability is called the *denial of service* attack. Note that ensuring the above elements of information security requires protection of multiple hardware and software components of the system, with the network being just one such a component.

Other major elements of information security include:

- *Authentication*—the ability of a party to verify the identity of the other communicating party or parties
- *Non-repudiation*—the ability to assert that a particular party generated the information.
- *Authorization*—the ability to restrict information access only to permitted parties
- *Authenticity*—the ability to verify that the information has been created or modified by the declared party
- *Privacy*—the ability to conceal the meta-data about the information-generating entity (e.g., identity of the entity, its location) that generated the transmitted data (to be distinguished from confidentiality)

As for most networks, WSNs are also expected to support a subset of the above information security goals, a set which depends on the particular WSN application. However, the new aspect of WSN security relates to the approaches through which the above information security goals are implemented. In particular, meeting the above security goals in WSNs are especially challenging, due to the hardware limitations of the sensor nodes (energy, computation, storage), as well as due to the operational modes of the network (e.g., unattended operation, large number of nodes, limited node lifetime, fixed deployment, unknown and/or changing propagation conditions). Furthermore, due to the fact that in some applications cost is an important consideration, individual network nodes tend to be inexpensive and, thus, less reliable. However, overall the network needs to support some required reliability level. Furthermore, the cost consideration often prevents reliance on more expensive hardware (e.g., tamper-resistant modules), which could otherwise reduce the complexity of some security-related protocols.

1.2 Implementation Constraints

The particular features of the sensor nodes, especially their limitations, affect the design of the security protocols. For instance, typically, nodes in a WSN are inexpensive sensing devices with quite limited computational capabilities, as compared with nodes in traditional communication networks. Consequently, WSN's nodes are unable to execute security protocols designed for such other networks. Furthermore, the wireless communication channel and the operational characteristics of WSNs introduce additional implementation challenges.

Hardware constraints

The complexity of a software and the performance requirements of its underlying

application dictate the amount of hardware resources needed for the software execution, including the size of memory/storage, the code space, the CPU clock rate, and the energy source. Sensors are usually limited in physical size, which in turn limits its ability to store data and code. Most contemporary sensor devices have RAM sizes ranging from 1K to 10K bytes, and program memory less than 1M bytes [3]. For example, a popular sensor type, the TelosB, uses the TI MSP430F1611 processor, which is a 16-bit, 8-MHz RISC CPU, with only 10KB RAM, 48KB program memory, and 1024KB flash storage. Moreover, the code space required to support the OS alone is on the order of several kilobytes. For example, the de-facto operating system for wireless sensors, the TinyOS, requires about 4KB of memory. Therefore, the implementation of security protocol must be very space-conscious to fit the code in the limited available memory of a sensor node.

Power constraints

Power constraints are always a major concern in WSNs, because the small physical size of the sensor node limits the battery size and, thus, its energy capacity. Furthermore, in many application scenarios in which sensor nodes operate unattended, once a sensor node is deployed it is impossible to replace its battery. Although recharging is possible if an energy scavenging mechanism is implemented in the nodes, often the amount of energy collected in this way is limited and inadequate to fully support by itself the node's energy needs. In contrast to the above power limitation, security related operations can be especially power demanding. Walters et al. summarized in [4] three major sources of power consumptions due to security related operations in WSN:

- the processing required for security functions, such as encryption, decryption, data signing, and signature verification;
- the energy required to transmit the security-related data or overhead, such as initialization vectors for encryption and decryption; and
- the energy required to maintain security parameters in a secure manner, such as storage of cryptographic keys.

Physical-level constraints

In general, wireless communication medium makes mounting security attacks much easier, as compared to attacks in wired networks. For example, the attacker can passively eavesdrop on the network's radio frequency range to steal messages in transit, or it can inject malicious messages into the network at will [5]. However, WSN are particularly vulnerable due to several of their attributes. The large number of deployed nodes and the lack of tight binding among the network nodes render it easier for an attacker to compromise a small number of nodes, while remaining undetected. Moreover, due to the simplicity of their hardware, sensor nodes will typically lack sophisticated protection schemes against physical-layer attack, such as jamming. Furthermore, in many WSNs, the senor nodes are unreliable and prone to hardware malfunction or depletion of energy, so that compromising a sensor node could be misinterpreted as a failure, rather than a security breach. To provide the required level of security, WSN security protocol design must take these physical-level constraints into consideration.

1.3 Potential Attacks and Defenses

Table 1: Sensor network layers and denial-of-service defenses. (From [6])

Network Layer	*Attacks*	*Defenses*
Physical	Jamming	Spread-spectrum, priority message, lower duty cycle, region mapping, mode change
	Tampering	Tamper-proofing, hiding
Data Link	Collision	Error-correcting code, Collision detection
	Exhaustion	Rate limitation
	Unfairness	Small frames
Network	Neglect and greed	Redundancy, probing
	Homing	Encryption
	Misdirection	Authorization, monitoring
	Black holes	Authorization, monitoring
Transport	Flooding	Client puzzles
	Desynchronization	Authorization

Wood and Stankovic in [6] exploited layered network architecture to analyze security issues and to improve robustness. The network layers architecture is divided into physical, data link, network, and transport layers. Each layer is susceptible to different types of attacks, and security attacks can exploit interaction among layers or cut across multiple layers. Table 1 lists the layers of a typical sensor network, and describes each layer's security vulnerabilities and possible defenses.

There are two major types of attacks at the physical layer: jamming and tampering [6]. Jamming refers to interfering with the transmissions on the radio frequencies that the network's nodes are using, such that an adversary can disrupt the entire network of N nodes with only k randomly distributed jamming nodes, where $k << N$. The standard defense against jamming involves various forms of spread-spectrum communication. However, such defense may not be available for sensor nodes, because sensor devices are typically assumed to be low-cost, low-power devices. Other defenses include switching to a lower duty cycle to outlive an adversary or mapping the jammed region and rerouting traffic. Tampering refers to physically compromising the nodes in the network. Tamper protection falls into two categories: passive and active [7]. Passive mechanisms do not require energy and include technologies that protect a circuit from being detected (e.g., tamper-proofing, protective coat). Active tamper protections involve special hardware circuits, which consumes more energy. Therefore, it would be more appropriate for sensor nodes to employ passive techniques.

The data link layer is susceptible to three major attacks: collisions, exhaustion, and unfairness [6]. Error-correcting codes can be used to alleviate some of the effects of collisions. However, they cannot completely solve the problem, as the adversary can still corrupt more data than could be corrected by the network. Collision detection is another way to deal with a collision attack, but it cannot completely defend against collision attacks, because proper transmission still need cooperation among nodes, while subverted

nodes could intentionally and repeatedly deny channel access. Exhaustion refers to attacks that deplete the energy source of the network nodes, thus jeopardizing the availability of the network. Existing MAC techniques, such as random back-offs and scheduled access, are only intended to solve the problem of random collisions. When collisions are intentional, these techniques become largely ineffective. A possible solution is inclusion of a rate limiting mechanism in MAC admission control, so that the network would simply ignore the excessive requests generated by an attacker without responding to such requests, and thus avoiding further increase in the traffic volume. However, rate limiting feature has its disadvantages; for example, it reduces the overall network capacity and it limits the maximum data rate of individual users even when the network is underutilized. Unfairness can be caused by selective intermittent application of the attacks mentioned above, by abusing a cooperative MAC-layer priority scheme, or by monopolizing the channel. One defense against this threat is to use small frames, so that an individual node can capture the channel only for a short time period. However, if the messages typically transmitted by the network nodes are long, then splitting the messages into smaller frames incurs additional framing and channel-access overheads.

The network layer is subject to four types of attacks: *neglect and greed*, *homing*, *misdirection*, and *black holes* [6]. A malicious node is *neglectful* when it arbitrarily neglects to route some messages. This node is also *greedy* if it gives undue priority to its own messages. A solution to this type of problem is to use multiple (alternate) routing paths or send redundant messages. Location-based network protocols that rely on geographic forwarding expose the network to *homing* attacks, in which an adversary observes the traffic to obtain the location of critical nodes. Once found, these nodes are subject to being attacked. One solution to this problem is to encrypt the header, so that an adversary cannot learn the location of the critical nodes from reading the header. This solution assumes a secure key management scheme, such that all neighbors share cryptographic keys, and so a passive adversary cannot learn the source or destination of the messages from the headers. *Misdirection* is a more powerful attack, which forwards messages along a wrong path, perhaps by fabrication malicious route advertisements. This attack can target either the sender or an arbitrary victim. (The defense to this attack is similar to that for a black-holes attack, which is discussed next.) A *Black-holes* attack is an even more effective attack against distance-vector-based routing protocols. In this attack, compromised nodes advertise zero-cost routes, thus forming *routing black holes* within the network [8]. This causes excessive messages to be routed through the compromised nodes, and therefore causes intensive bandwidth contention around malicious nodes. It also causes the neighbors of malicious nodes to quickly exhaust their energy supplies due to excessive routing, and therefore could potentially create partitions in the network. Authorization and monitoring are ways to defend against misdirection and black-holes attack [6]. In an authorization-based solution, only authorized nodes (i.e., nodes with valid public/private key pairs) are allowed to exchange routing information. Nodes may use public key infrastructure to sign and verify routing messages, thus ensuring the confidentiality and integrity of routing information. Zero-cost routes in black holes attack are thus eliminated, as an adversarial node does not have the valid public/private key pairs to generate encrypted routing information. Such a scheme requires a reliable certification authority for authentication. Since a centralized certification authority can become a single point of failure, distributed certification

authority schemes have been proposed. For example, Zhou and Haas in [9] proposed a distributed certification authority scheme by distributing the certification function among *n* servers and, by using threshold cryptography, ensures the certification authority is compromised only if at least *t* servers are compromised. Monitoring-based solution relies on nodes monitoring their neighbors to ensure their proper routing behavior. Nodes then select routing paths utilizing nodes that exhibit long-term proper routing behavior. It is assumed in monitoring-based solution that nodes with a longer period of proper routing behavior are more trustworthy, and therefore routing information from these nodes are less likely to be inaccurate. This defends misdirection and black-holes attack, as compromised nodes can be quickly detected by their neighbors, and the compromised nodes are then not selected again for routing paths.

The transport layer can be threatened by flooding and desynchronization attacks [6]. A naïve solution to *flooding* is to limit the number of allowed connections, but this would also degrade the overall network throughput, as there would be fewer connections available for each node. A better solution is to have the server node ask the client node to solve some computationally expensive puzzle upon requesting a connection, so that if the client is a compromised node and repeatedly requests establishment of connections, the client would deplete its power while repeatedly solving the puzzles. *Desynchronization* disrupts end-to-end connections. In this attack, the adversary disrupts the communication between two nodes by forging messages that carry sequence numbers or other control messages. If the adversary is successful, the communicating nodes will waste energy by excessively executing the synchronization-recovery protocol without exchanging any useful information. A counter to this attack is authentication of all exchanged packets, assuming that the adversary cannot forge the authentication mechanism.

1.4 Evaluation Benchmarks

There are various benchmarks for evaluating whether a security scheme is appropriate for WSNs. Some of these benchmarks are ([10]):

- Resiliency—the network capability to continue to offer sufficient security services while some of the network nodes are compromised.
- Resistance—the network capability to avert an attacker from fully controlling the network.
- Scalability—the capability to support security in very large networks (on the order of hundreds to thousands of nodes).
- Self-organization and flexibility—the capability to adaptively restructure the security scheme as a result of network changes;
- Robustness—the capability of the network to continue to operate in spite of irregularities (e.g., security attacks, hardware failure, etc). (Robustness is a more generalized notion than resiliency, as resiliency is focused on providing security services while under attack, whereas robustness considers providing general network services while under attack.)
- Energy efficiency—the degree to which the network lifetime is maximized by preserving energy.
- Assurance—the confidence that the major elements of information security (e.g., confidentiality, integrity, availability, etc.) are adequately met [2].

2. Basic Security Challenges and Approaches

2.1 Cryptography Schemes

Cryptography is the basic element in any security system. It deals with encrypting the message in order to achieve secure communication in the presence of third party adversaries. There are two types of encryption methods: symmetric key cryptography, which uses the same key for encryption and decryption, and asymmetric (public) key cryptography, which uses different keys for encryption and decryption. Compared to asymmetric key cryptography schemes, symmetric key cryptography schemes have the advantage of lower computational overhead because they involve relatively simply bit-wise operations (e.g., XOR) that can be directly implemented in hardware, but they require more complex key distribution and key management schemes ([11]).

There has been extensive work done on the evaluation of different cryptography schemes. In [12], Ganesan, et al. investigated the performance of five different symmetric key cryptography schemes (RC4, IDEA, RC5, MD5, and SHA1), over six different hardware platforms that use 8/16/32-bit word size (Atmega 103, Atmega 128, M16C/10, SA-1110, PXA250, and UltraSparc2). Their experiments indicate that: (1) the cycle overhead (i.e., the number of clock cycles to perform a cryptographic operation on a hardware platform) is mostly uniform within each word-size class (8/16/32 bit), but there are differences among the three word-size classes; (2) the impact of caches (i.e., the additional clock cycles due to cache misses in memory fetch) is negligible; and (3) hashing techniques require almost an order of a magnitude higher clock cycle overhead than symmetric key encryption techniques. Table 2 shows the execution times for the various encryption algorithms on the various platforms. Figure 1 shows the byte overhead for the various algorithms and platforms. They also derived a model to assess the computational overhead of embedded architectures for encryption protocols, in general.

Table 2: Execution times (in µs) for algorithms, platforms, and plaintext sizes (in bytes). (From [12])

Algorithm	Size	Action	Atmega 103	Atmega 128	M16C/10	strong ARM	Xscale (400)	Xsacle (200)	Sparc (440)
MD5	0	Digest	5863	1466	1083	46	26	53	23
	1-26	Digest	5890	1473	1075	46	26	53	23
	62-80	Digest	10888	2722	2011	74	45	90	39
SHA-1	1	Digest	15249	3812	2651	69	12	102	27
	3	Digest	15781	3945	5303	69	12.3	103	27
	56	Digest	14543	3636	7955	133	25.8	205	55
	64	Digest	31107	7777	10907	145	25.7	207	56
RC5	16	Init	9641	2410	2074	41	45	91	28
		Enc	1651	413	197	3	3	6	2
		Dec	1636	409	202	3	3	7	2
IDEA	16	Init enc	1523	381	727	26	15.54	47	11
		Init enc	9417	2354	1927	76	25.16	69	36
		Enc	2555	325	596	16	3.24	17	9
		Dec	2614	325	597	16	3.27	17	9
RC4		Init	1886	472	2455	155	66.8	216	96
		Enc	344	86	123	10	5	9	4

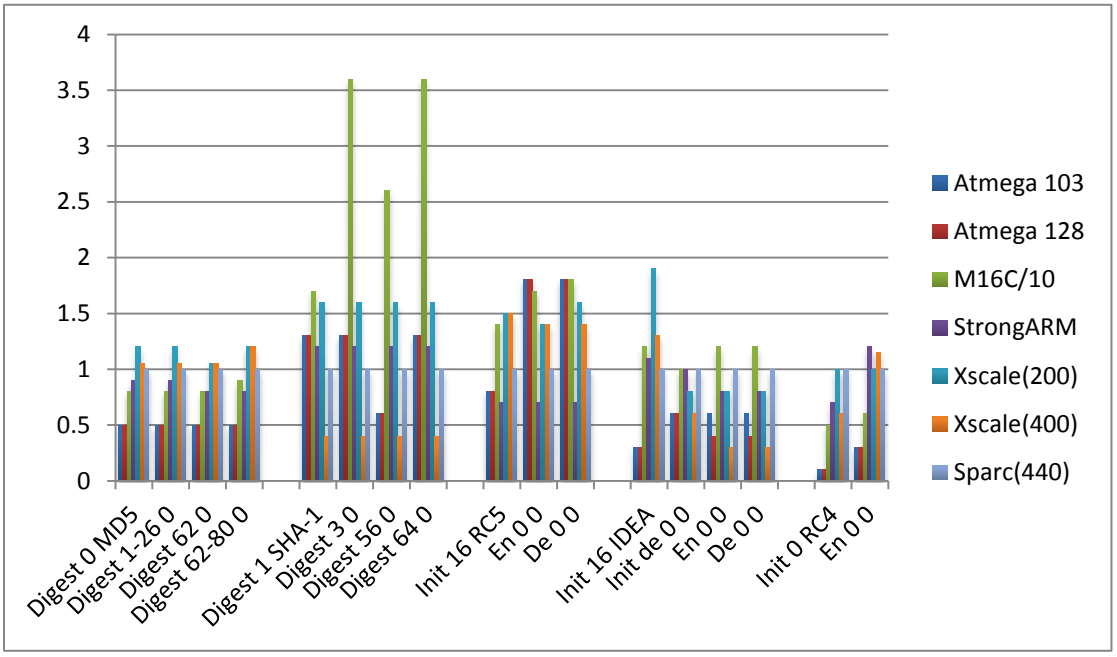

Figure 1: Normalized overhead for algorithms, platforms and, plaintext sizes (in bytes). (From [12])

Although public key cryptography schemes have received much less attention in WSN security due to their expensive computational overhead, there are several studies that discuss the possibilities of incorporating public key cryptography schemes into WSN. In [13], Gaubatz, et al. showed that special purpose ultra-low power hardware implementations of public key algorithms can be used in sensor nodes. They implemented three different public key cryptography schemes (Rabin's Scheme [14], NtruEncrypt and NtruSign [15], and Elliptic Curve Cryptography [16, 17]) in a sensor node that is embedded with a custom-designed low-power co-processor to handle all the computation-intensive tasks. They concluded that the use of public key cryptography can reduce the amount of traffic overhead due to key management in WSN, and that the computational cost is within acceptable limits and sufficiently fast on their special-purpose hardware. Wander et al. in [18] also performed a series of experiments to quantify the energy costs of authentication and key exchange based on ECC [16, 17] and RSA [19] public key cryptography schemes on an 8-bit microcontroller platform. Their studies indicate that authentication and key exchange protocols using optimized software implementations of public key cryptography are quite viable on small wireless devices. They also recommend ECC over RSA for larger energy savings.

Another cryptography technique is watermarking. In [20], Koushanfar and Potkonjak proposed the first watermarking approach for protecting data and information generated in wireless embedded sensor networks. They considered a sensor network application in which sensor nodes collect data (the data-acquisition phase) to solve nonlinear optimization problem (the data-processing phase). Node signatures are embedded during the data-acquisition and data-processing phases, without compromising the quality of the recorded data or the results od data processing. They conducted two experiments—

acoustic atomic trilateration and light source determination—to study the trade-offs between the security protection and the watermarking overhead and to study the situations where such watermarking scheme is the most effective.

2.2 Key Management Schemes

Key management deals with distributing and storing encryption and decryption keys to implement secure communication. A trivial solution to key management is to use a global key for all the sensor nodes. However, in this scheme, if any node in the network is compromised, then the adversary obtains the global key and the whole network security is defeated. Another trivial solution is to have each node store N-1 different keys (where N is the total number of nodes in the network), with each key corresponding to a different node in the network. However, this solution is too complex, as a sensor node's limited memory may be insufficient to store the N-1 keys, especially for a large network. Clearly, key management is an important yet challenging task, in particular for encryption schemes that use symmetric key cryptography.

There have been extensive research works done in the area of key management schemes. Here, we discuss four categories of key management schemes—key pre-distribution schemes, hybrid cryptography schemes, key infection schemes, and key management in hierarchy networks. Figure 2 shows the taxonomy of key management.

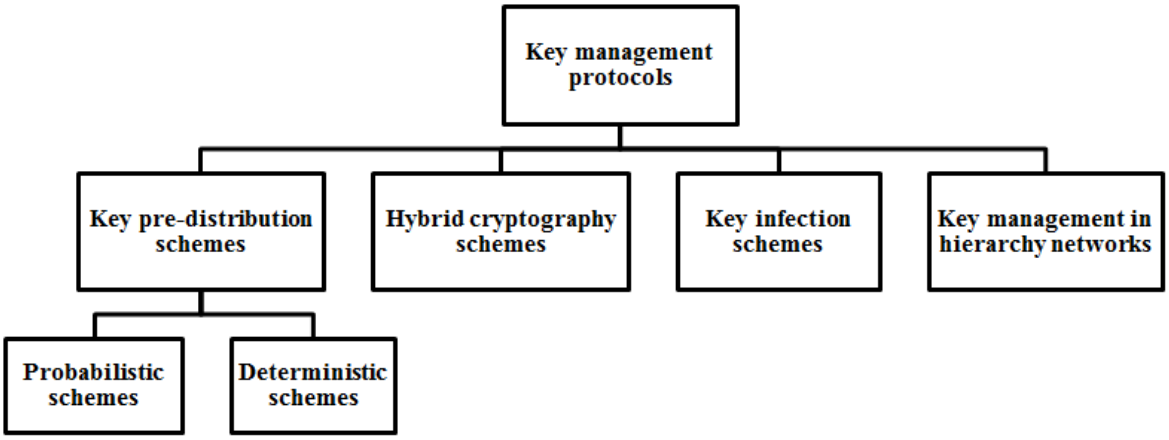

Figure 2: Taxonomy of key management protocols

2.2.1 Key pre-distribution schemes

In key pre-distribution schemes, sensor nodes store some initial keys before the nodes are deployed [21]. Key pre-distribution schemes are further divided into probabilistic schemes and deterministic schemes.

For probabilistic schemes, the existence of one or more common predistribution keys between intermediate nodes is not certain, but is instead guaranteed only probabilistically. Eschenauer and Gligor in [22] proposed one of the earlier probabilistic schemes. In their scheme, a ring of keys is distributed to each sensor node before node deployment. Each key ring consists of a randomly chosen k keys from a large pool of P keys, which is

generated off-line. A pair of nodes can communicate if they share any key among their key rings. Although a pair of nodes may not always have a shared key, if a path between them exists, they can use that path to exchange a key that establishes a direct link. An enhancement over this scheme is proposed by Chan, et al. in [23], in which a *q*-composite random key pre-distribution scheme is proposed. This scheme requires q keys (q>1) instead of just 1 common key among the key rings of a pair of communicating nodes. The authors showed that the *q*-composite key scheme strengthens the network's resilience against node capture when the number of captured nodes is small. Figure 3 shows how the fraction of additional communications that the attacker can compromise varies with the number of nodes captured by the attacker. As a point of reference, comparing the two cases of q = 1 and q = 2, in terms of the amount of additional compromised communications in a network with 50 compromised nodes is 9.52%, and 4.74%, respectively. The disadvantage of the q-composite keys schemes is that a larger portion of the network is revealed to the adversary as larger number of nodes becomes compromised. This scheme thus trades off the protection against an unlikely large-scale network attack in order to significantly improve the strength of the random key pre-distribution scheme against smaller-scale attacks. Another probabilistic scheme is GKMPSN proposed by Zhu and Zhang in [24]. GKMPSN is a centralized group key distribution scheme, in which a network controller broadcasts new group keys, as well as node revocation information (i.e., information that identifies a compromised node), to all the nodes whenever a compromised node is detected. Prior to the deployment of the network, each node stores a random set of keys out of a common large key pool. The group re-keying operation then takes two steps. In the first step, the pre-deployed random keys at each node are used to create secure channels between nodes in order to deliver new keying materials to legitimate nodes. In the second step, each node uses the received keying materials to update both the group key and the pre-deployed keys that are invalidated by the compromised nodes. GKMPSN has an attractive property of partial statelessness, in which a node can decode the current group key, even if the node missed a few previous group re-keying operations. This is an attractive feature as: (1) typically packet losses are high in WSN due to unreliable communication, and (2) the scheme facilitates new nodes joining the network after initial network deployment.

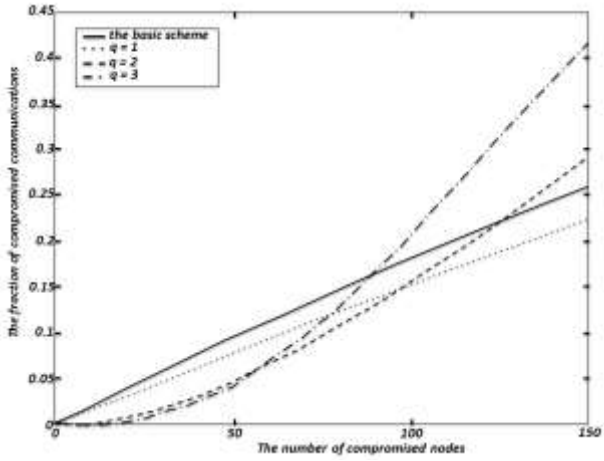

Figure 3: Probability that a random communication link between two randomly chosen nodes can be decrypted by an adversary, as a function of the number of nodes captured

by the adversary (excluding the two communicating nodes). Key ring size is 200, and probability of successful key-setup with a neighbor is 0.33. (From [23])

For deterministic schemes, any two intermediate nodes are guaranteed to share one or more predistributed keys. An example of a deterministic scheme is LEAP (Localized Encryption and Authentication Protocol) proposed by Zhu, et al. ([25]). LEAP is motivated by the observation that different types of messages have different security requirements and that a single keying mechanism is not suitable for meeting these different security requirements. LEAP supports the establishment of four types of keys for each sensor node—an individual key shared with the sink node, a pairwise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a group key that is shared by all the nodes in the network. The individual key allows a node to securely send sensor readings to the sink node. The pairwise key prevents a compromised node's attack, because once a compromised node is detected, its neighbors will typically immediately revoke the pair-wise keys shared with that compromised node. However, even if the compromised node is not detected for some time, its damage is only limited to its near neighbor, as the pair-wise keys are only shared between one-hop neighbors. The cluster key is used for secure local broadcast, for example routing control information. The group key authenticates the sink node to the sensor nodes to facilitate secure network-wise operations, such as key refreshments. Blom in [26] proposed another pair-wise deterministic key distribution scheme. The scheme can defend against up t compromised nodes. In pre-distribution phase the sink node generates a (t+1)-by-N matrix $\mathcal{G}$ over some finite field GF(q), where N is the total number of nodes in the network and (t+1) is the codeword size. The matrix $\mathcal{G}$ is known to all the nodes in the network. Then the sink node creates a (t+1)-by-(t+1) symmetric matrix $\mathcal{D}$ over GF(q). This allows the sink node to compute a matrix $\mathcal{A} = (\mathcal{D}\mathcal{G})^T$ with the property that $\mathcal{K} = \mathcal{A}\mathcal{G}$ is a symmetric matrix (since $\mathcal{A}\mathcal{G} = (\mathcal{D}\mathcal{G})^T\mathcal{G} = \mathcal{G}^T\mathcal{D}^T\mathcal{G} = \mathcal{G}^T\mathcal{D}\mathcal{G} = \mathcal{G}^T\mathcal{A}^T = (\mathcal{A}\mathcal{G})^T$). Each node i in the network is assigned with a public column vector $\mathcal{G}^{(i)}$=i^{th} column of $\mathcal{G}$, and a private row vector $\mathcal{A}(i)$=i^{th} row of $\mathcal{A}$. Then, for nodes i and j to establish a pairwise key, they can exchange $\mathcal{A}(i)$ and $\mathcal{A}(j)$, and compute their pairwise key $K_{ij} = K_{ji} = \mathcal{G}^{(j)}\mathcal{A}(i) = \mathcal{G}^{(i)}\mathcal{A}(j)$. Assuming there are $m < t+1$ compromised nodes, then these nodes know m rows and, due to symmetry, m columns of $\mathcal{K}$. A node needs to know at least (t+1) elements in a codeword in order to acquire information about other element in the codeword of other nodes. Therefore, if there are less than (t +1) cooperating nodes, no information about the unknown key is revealed. It has been proven in that the Blom scheme securely protects the pairwise key if any t+1 columns of $\mathcal{G}$ are linearly independent.

2.2.2 Hybrid cryptography schemes.

Hybrid cryptography schemes use computationally expensive asymmetric key cryptography at the sink nodes and computationally cheaper symmetric key cryptography at all other sensor nodes. An example of hybrid scheme is proposed by Huang, et al. in [27], which is based on a combination of elliptic curve cryptography and symmetric key operations. This scheme reduces the high cost of elliptic curve random point scalar multiplications at the sensor side and replaces them with low cost and efficient symmetric key based operations. On the other hand, it authenticates the two identities based on

elliptic curve implicit certificates to avoid the typical key management problem in pure symmetric key based protocols.

2.2.3 Key infection schemes

In key infection schemes, keys are sent in plaintext and thus are not secure. However, these schemes assume that the number of adversaries at key establishment phase is very small. For example, Anderson, et al. in [28] proposed a scheme in which each node bootstraps itself by broadcasting an initial key in the clear. Nodes then exchange keys and build up trust structures as they perform network and resource discovery. The scheme assumes that the adversary can only monitor a small proportion of the communications during deployment phase (i.e., initial key setup phase), but is fully capable of launching attacks after the deployment phase. This is often a realistic assumption, because the initial deployment duration is on the order of seconds, while the overall lifetime of the network can be up to years. Despite the apparent insecurity of this scheme, the proposed scheme uses *multipath secrecy amplification* and *multi-hop key propagation* to enhance the security of the network, such that at most a fixed proportion of communications links can be eavesdropped. *Multipath secrecy amplification* combines keys propagated along different paths to update pairwise keys. For example, consider three nodes W_1, W_2, and W_3, with pairwise keys k_{12}, k_{13}, and k_{23}. Suppose W_1 wants to update k_{12} to k_{12}'. W_1 can ask W_3 to send the key-update request to W_2. The request from W_1 to W_3 is encrypted using k_{13}, and the request from W_3 to W_2 is encrypted using k_{23}. Therefore, if k_{12} is the only key being compromised, the adversary cannot update to a new k_{12}' as long as neither k_{23} nor k_{13} is compromised. Table 3 compares the ratio of the compromised links of the two schemes: the basic key infection scheme and the security amplification scheme (SA), showing the improvement of the latter scheme. In the table, α is a varying density of the adversarial nodes (referred to as "black dust"), assuming values of 1%, 2%, and 3%, and *d* is the average number of neighbors of a node. *Multi-hop key propagation* uses intermediate nodes to temporarily store the pairwise key update information for two nodes at the ends of a path. For example, if W_1 links to W_2, W_2 links to W_3, and W_1 wants to update k_{13} to k_{13}', then W_1 and W_3 can invoke W_2's help to set up a new key that W_2 immediately forgets, so a potential node compromising W_2 in the future does not reveal k_{13}'. Multi-hop key propagation supports end-to-end, rather than link-level cryptography, which helps energy efficiency as sink-to-node communications can be encrypted using end-to-end keys rather than translated at intermediate nodes. With the assumption of limited adversaries at the initial key deployment phase, and with the enhancement using multipath secrecy amplification and multi-hop key propagation, the simulation showed that the key infection scheme is almost as secure as using pre-loaded initial keys.

	$\alpha = 1\%$		$\alpha = 2\%$		$\alpha = 3\%$	
d	**basic**	**SA**	**basic**	**SA**	**basic**	**SA**
2	1.20%	0.97%	2.29%	2.00%	3.38%	2.93%
3	1.81%	1.37%	3.44%	2.67%	5.42%	3.93%
4	2.30%	1.80%	4.45%	3.71%	6.50%	5.55%
5	2.93%	2.37%	5.73%	4.68%	8.73%	6.75%

Table 3: Improvement of secrecy amplification (SA) over the basic key infection scheme (From [28])

2.2.4 Key management in hierarchy networks

Some key management schemes take advantage of the fact that nodes are often categorized into different types, such as sink nodes, gateway nodes, and sensor nodes, and different types of nodes have different computational resources. In [29], Jolly et al. present a key management scheme in a clustered sensor network. The method uses pre-deployed symmetric keying, in which sensor nodes store a minimum number of keys that they share with other nodes. Gateway nodes store a larger number of keys, and the sink nodes have no restrictions and store all the keys in the network. Their simulation showed that the energy consumption overhead for the key management is remarkably low and they report an order of magnitude of energy saving. Chorzempa, et. al., in [30] proposed another hierarchical key management scheme for WSNs. The scheme is called SECK (Survivable and Efficient Clustered Keying), with three tiers of nodes. The bottom tier consists of low-end sensor nodes, which are clustered. Each cluster is managed by a second-tier cluster head to perform data aggregation and forwarding. At the top tier there is a globally trusted sink node. After initial network deployment, the low-end sensor nodes undergo a location training phase to establish clusters, and cluster coordinate system is used in low-end node recovery procedure. Clusters are then used for establishing and updating administrative keys. A session key between a pair of nodes can then be obtained from administrative keys. Simulations suggested that the scheme is resilient against multiple node captures, and can efficiently recluster and salvage compromised nodes. Figure 4 depicts the comparison of the resiliency against node capture of the SECK scheme and the basic probabilistic pairwise scheme of Eschenauer and Gligor [22], while showing that the resiliency of both schemes is comparable. Realizing that the Eschenauer and Gligor scheme is considered as having good resiliency, one can conclude that the SECK scheme has overall good resiliency property as well.

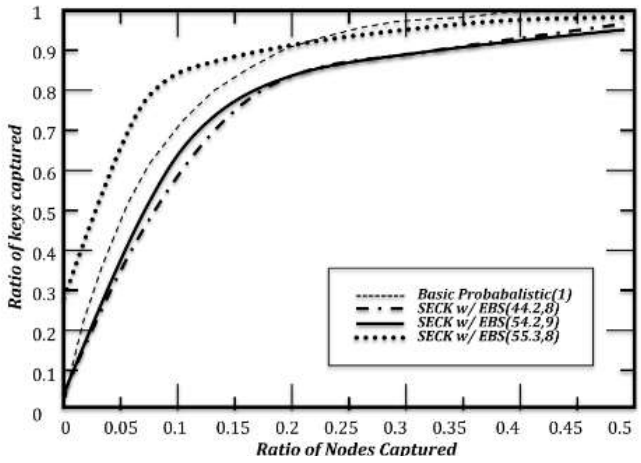

Figure 4: The ratio of keys captured vs. the ratio of the captured nodes. (From [22])

2.3 Attack Detection and Prevention

Section 1.3 briefly discussed several basic attacks and defenses in WSNs. This section extends the previous discussion and focuses on attack detection and prevention mechanisms for two well-known attacks in WSNs: the Sybil attack and the Wormhole attack. Another, a more complex issue, compromised node detection, is discussed here as well.

2.3.1 Sybil attack detection and prevention

Newsome et al. systematically analyzed the Sybil attack and its defensive measures in [31]. In the Sybil attack, a node illegitimately claims multiple identities. This attack can be exceedingly detrimental to many important functions of WSN. Figure 5 demonstrates Sybil attack, where an adversary node 'AD' is present with multiple identities. 'AD' appears as node 'F' to 'A', as node 'C' to 'B', and as node 'A' to 'D', so when 'A' wants to communicate with 'F', it sends the message to the adversarial node 'AD'.

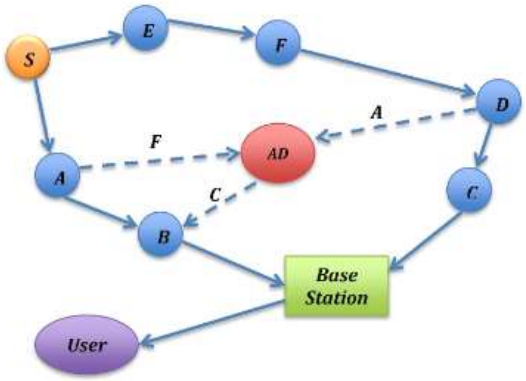

Figure 5: Sybil Attack

In distributed storage for WSNs, the Sybil attack can defeat replication and fragmentation mechanisms in a distributed hash table, such as GHT [32]. Thus, the system may not realize that while it replicates or fragment data across a number of nodes, in fact, it is storing data on a number of Sybil identities which were created by the same malicious node. For routing, the Sybil attack can defeat multipath or dispersity routing protocols, such that seemingly disjoint paths could, in fact, traverse through a single malicious node which represents several Sybil identities. In geographic routing protocols, a Sybil node could appear as being present in more than one place at the same time. For data aggregation, the Sybil attack can have one malicious node contribute to the aggregate many times to alter the aggregate reading. For voting, the Sybil attack allows a malicious node to vote multiple times to control outcome of a vote, such as in a blackmail attack. For fair resource allocation, a Sybil node can claim multiple identities and therefore obtain more network resources. For misbehavior detection, Sybil nodes could "spread the blame" by making it appear that the level of misbehaving of the Sybil identities is large enough for the system to take an action. Defenses against the Sybil attack include radio resource testing, random key predistribution, position verification, and registration [31]. In radio resource testing [31], it is assumed that any sensor node has only one radio, and that a radio is incapable of simultaneously sending or receiving on more than one channel. When a node *A* wants to test whether any of its neighbors are Sybil nodes (i.e., a node with multiple identities), the node *A* can assign to each of its neighbors a different channel to broadcast some messages. The node *A* can then choose randomly a channel to listen to. Due to the assumption that each node has only one radio, if *A* does not hear anything on a chosen channel, then the node *A* can be suspect that the node being assigned to that channel is a Sybil node. Figure 6 shows the probability of not detecting the presence of

some Sybil nodes using this method.

In random key predistribution [31], each node is assigned a subset of a large set of keys, such that any two nodes share at least a secret key for communication, and no two nodes are assigned the same subset of keys. As a result, the sensor node can be uniquely identified by the subset of keys that it possesses. A network is able to verify the identity of a node by the keys that the node possesses, referred to as *key validation*. To launch a successful Sybil attack, the attacker is challenged to find the exact subset of keys of a node to steal the node's identity. The estimated probability of a randomly created Sybil identity being effective is depicted in Figure 7 as a function of the number of compromised nodes. As a point of reference, for an attack to succeed, the attacker needs to compromise at least 150 nodes (in the full validation case).

Position verification is another approach to defend against Sybil attacks [31]. This approach is only applicable in static WSNs (i.e., where sensor nodes are not mobile). In this approach, the network verifies the positions of each node. Sybil nodes can be detected because the Sybil nodes advertised by a single malicious node now all have the same physical position, which would raise an alarm, as the assumption is that a single physical location could be associated with at most one node.

Registration is another potential solution against Sybil attacks [31]. In this approach, there is a trusted central authority that manages the network. The central authority keeps a list of trusted nodes and deployment of nodes. In order to detect a Sybil attack, an entity would poll the network; the results would then be compared with the information about the known deployment. To prevent attack, any node could query the central authority for checking the trusted node list. This scheme requires that the central authority must be able to store the trusted node list and deployment information securely.

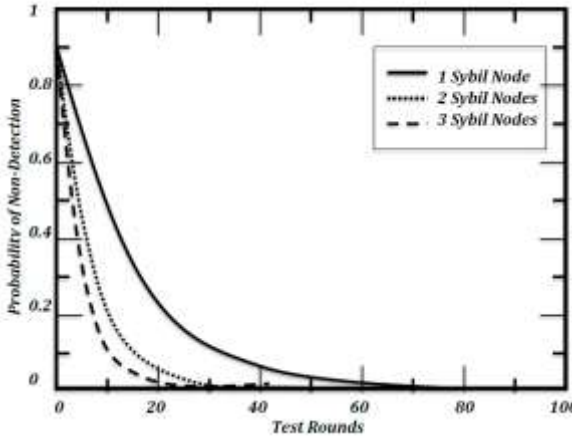

Figure 6: Probability of no Sybil nodes being detected while using the radio defense in the case of a channel being assigned to every neighbor (From [31])

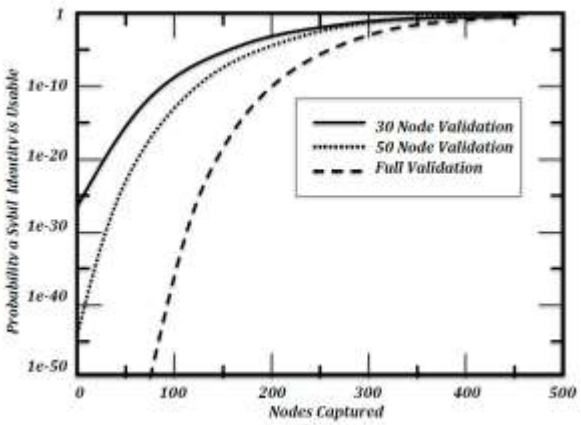

Figure 7: Probability of a randomly created Sybil node being effective in the key pool scheme as a function of the number of compromised nodes (From [31])

2.3.2 Wormhole attack detection and prevention

In the wormhole attack, an attacker records packets at one location in the network, tunnels them to another location, and retransmits the packets at the other location, making it appear as the two parts of the tunnel are in close proximity to each other. Figure 8 demonstrates the wormhole attack, where 'WH' is the adversary node which creates a tunnel between nodes 'E' and 'I'. These two nodes now apprear as they are at most at the distance of two hops from each other.

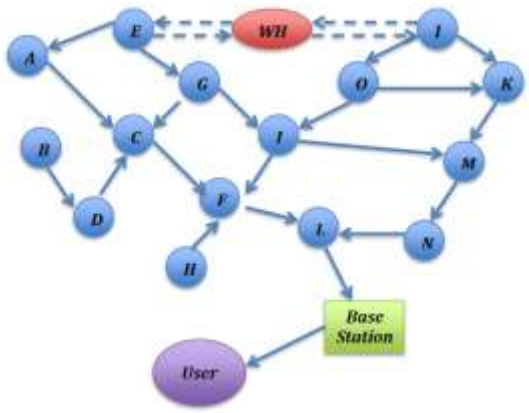

Figure 8: The Wormhole Attack

The wormhole attack can form a serious threat in wireless networks. For example, the wormhole attacker can gain unauthorized access, disrupt routing, or perform a Denial-of-Service attack. In [33], Hu et al. described the wormhole attack and proposed a mechanism called *packet leashes* for detecting and defending against wormhole attacks. A leash is an additional part of a packet that limits the packet's maximum allowed transmission distance. The authors introduced two types of leashes—geographic leash, which limits the distance a packet travels, and temporal leash, which limits the time a packet lives (and hence limits the travelling distance, as packet's speed is limited by its

speed of propagation). To construct a geographical leash, each node is assumed to know its own position, and all nodes in the network are assumed to have loosely synchronized clocks. Consider two nodes: source, *s*, and receiver, *r*. Let *ps*, *pr*, *ts*, and *tr* denote positions and times of nodes *s* and *r*, respectively. If *v* is an upper bound on the velocity of any node, and Δ is the maximum clock difference between any two nodes, then upon receiving a packet from source *s*, the receiver can compute an upper bound on the distance between the sender and itself, as $dsr \leq \| ps - pr \| + 2v\,(tr - ts + \Delta) + \delta$, where δ is the maximum relative error in location information between any two nodes. To use temporal leashes, the packet includes an expiration time, after which the receiver does not accept the packet. The expiration time is based on the allowed maximum transmission distance and the speed of light. A specific protocol, called TIK (TESLA with Instant Key disclosure), is also presented in [33] to implement temporal leashes. TIK consists of three states: *sender setup*, *receiver bootstrapping*, and *sending and verifying authenticated packets*. In the *sender setup* phase, the sender uses a pseudo-random function *F* and a secret master key *X* to derive a series of keys $K_0, K_1, \ldots, K_w$, where $K_i = F_X(i)$. The pseudo-random function is assumed to be secure in the sense that it is computationally intractable for an attacker to find the master secret key *X*, even if all the keys $K_0, K_1, \ldots, K_w$ are known. In addition, without the secret master key *X*, it is computationally intractable for an attacker to derive a key K_i that the sender has not yet disclosed. Each K_i expires after some time interval *I*, which is selected by the sender. In the *receiver bootstrapping* phase, the receiver synchronizes with the source to agree on the initial time T_0 and the time interval *I*. Finally, in the *sending and verifying authenticated packets* phase, if the sender sends a packet *P* at time T_i, then the sender also sends a message authentication code (HMAC) of *P* generated using some undisclosed key K_{i+j}, in which *j* is large enough such that *P* arrives at the receiver before time T_{i+j} and K_{i+j} has not yet been disclosed. The receiver can wait until time T_{i+j} for the source to release the key K_{i+j}, and verify the HMAC for *P*. The timing diagram of a TIK packet is shown in Figure 9, where τ is the propagation time between the nodes. The protocol assumes that all the clocks are synchronized within the maximum timing error of Δ. Upon receipt of the HMAC value and based on the time T_i as the time of disclosure of the key K_i, the receiver confirms that the corresponding key K_i was not yet sent by the sender. After all the verifications of the protocol were successfully completed, the receiver accepts the packet.

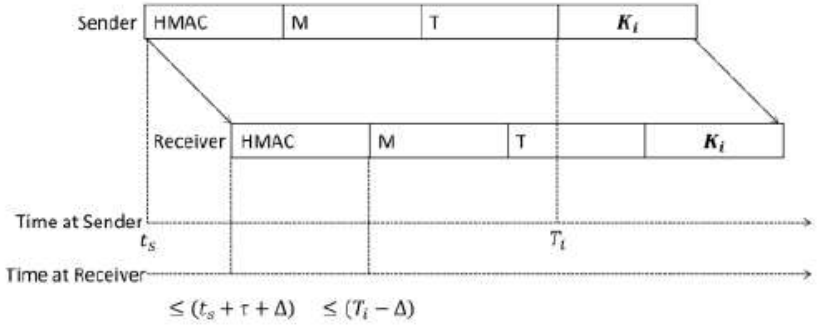

Figure 9: Timing of a packet in transmission of the TIK protocol. (From [33])

2.3.3 Compromised node detection

Besides attack detection and prevention, compromised node detection is another important problem in WSN security. Compromised node detection is usually implemented by software or hardware code-testing schemes. However, for WSNs, hardware-based code-testing schemes are often not feasible for lightweight sensor nodes. Software-based code-testing is more promising, because it requires neither dedicated hardware nor physical access to the device. Software-based approaches are usually based on a challenge-response scheme, where the verifier (usually the sink node) challenges a prover (a target device) to compute a checksum of its memory [34]. Examples of software-based code-testing schemes are SWATT [35] and SCUBA [36].

SWATT (SoftWare-based ATTestation for Embedded Devices)[35] by Seshadri et al. is a software-based attestation technique to verify the memory contents of embedded devices. For each sensor device, SWATT adds an external verifier that is physically distinct from the device. The verifier and the device then run the challenge-response protocol of SWATT. To ensure that the device can return the correct answer only if its memory contents are correct, the verification procedure uses a pseudorandom memory traversal, in which the verifier sends to the device a randomly-generated seed for the device to generate a pseudorandom starting memory address for the verifier to access. The verifier then traverses the memory randomly, and iteratively updates a checksum of the memory contents. Since the verifier's memory traversal is random, the attacker cannot predict which memory location is accessed, and therefore after a certain number of iterations of memory accesses, the verifier can eventually detect whether the memory is maliciously altered.

SCUBA (Secure Code Update By Attestation) [36] by Seshadri et al. is another example of software-based code-testing scheme. SCUBA enables a sensor network to detect compromised nodes. Once a compromised node is detected, SCUBA allows the network to either repair the compromised node through code updates, or revoke the compromised node. SCUBA is based on ICE (Indisputable Code Execution), which is a challenge-response-based protocol that ensures that a remote sensor node does not execute a malicious executable. To achieve this, each sensor node is installed with a special executable called the ICE verification function. The ICE verification function is responsible for checking the integrity of an executable on the sensor node, and also for setting up an execution environment to provide atomic execution for this executable (i.e., when an executable is executed in this execution environment, no other executable can interrupt this execution). To ensure that the ICE verification function itself is untampered, the ICE verification function is implemented as a self-checksumming code, which is a sequence of instructions that compute a checksum over themselves, such that the checksum would be wrong or the computation would be slower if the sequence of instructions were modified. The SCUBA protocol then works as follows. To invoke a sensor node's executable X, the invoker node (e.g., a sink node) first sends a "check integrity and execute" request to the sensor node. The ICE verification function on the sensor node then checks for the integrity of X, and executes X if the integrity checking passed. After finishing the execution of X, the sensor node sends the execution result, together with the checksum returned by the ICE verification function, back to the invoker. The invoker can detect whether the sensor node's executable is compromised by reviewing the ICE checksum.

3. Secure Routing

Many WSN routing protocols are based on traditional ad hoc network routing protocols. The original focus of ad hoc routing protocols was on performance, but security issues were extensively studied as research on ad hoc routing protocols matured (e.g. [37, 38, 39, 40, 41, 42]). Of course, secure routing is also an important requirement for WSN applications. This section examines existing approaches to secure routing for WSNs. Figure 10 shows the taxonomy of secure routing. A comprehensive discussion of many of the attacks on routing protocols is also presented in [43] by Karlof et al.

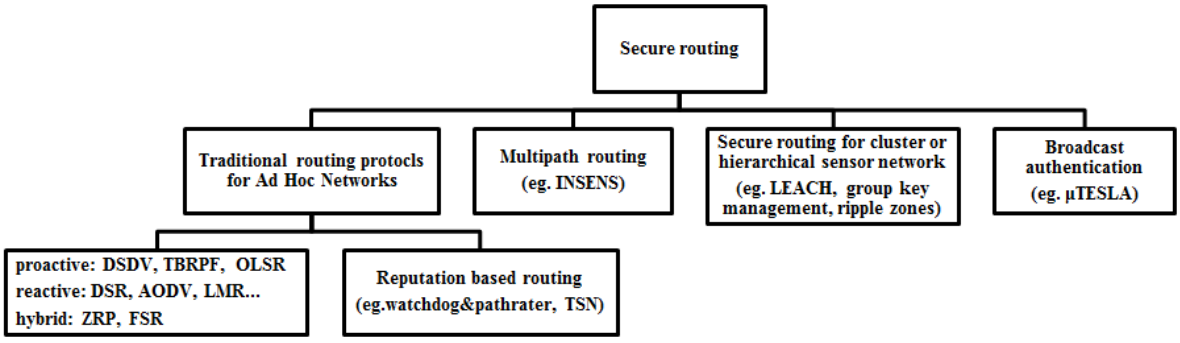

Figure 10: Taxonomy of Secure Routing

3.1 Traditional Routing Protocols for Ad Hoc and Sensor Networks

Routing protocols for ad hoc and sensor networks can be broadly classified into three categories: proactive, reactive, and hybrid [44]. In proactive routing protocols, nodes periodically exchange routing information, so typically correct routes are almost always known at the time a routing request is placed. Examples of proactive routing protocols include DSDV[45], TBRPF[46], and OLSR[47]. In reactive routing protocols, nodes exchange routing information only when a communication request is pending. Examples of reactive routing protocols include DSR[48], AODV[49], LMR[50], ABR[51] and TORA[52]. Hybrid routing protocols are a mixture of proactive and reactive routing protocols. ZRP [53] and FSR[54] are examples of hybrid routing protocols. Proactive routing protocols have lower latency, since routing information is consistently maintained; however such protocols are wasteful in communication overhead when the traffic activity is small and, especially, when the network is highly mobile. In contrast, reactive routing protocols incur smaller communication overhead at the expense of larger delays. Thus, reactive protocols may be more practical for low-activity and mobile ad hoc networks, while proactive protocols may better fit highly-active and more static networks. Similarly, for mobile sensor networks which support applications that require infrequent communications, reactive protocols might be a better choice. Hybrid routing protocols typically outperforms proactive and reactive routing protocols, because they use a combination of the two [55], and often optimize their performance based on the network conditions [53,56].

The original designs of many of the ad hoc network routing protocols are based on performance metrics (e.g., energy efficiency) rather than on security provisions, but there have been numerous works that extend these original ad hoc network routing protocols to

improve security. Zapata in [37] noted that ad hoc networks protocols are being designed without security in mind. He proposed the secure ad hoc on-demand distance vector (SAODV) routing protocol to address the problem of securing a MANET network. SAODV assumes that each node has a signature key pair from a suitable asymmetric cryptosystem. Further, each ad hoc node is capable of securely verifying the association between the address of a given ad hoc node and the public key of that node. AODV uses two mechanisms to secure its messages: Digital signatures and Hash chains. Digital signatures are used by AODV to authenticate the non-mutable fields of a message. Hash chains are used in AODV to secure the mutable part of a message, which is hop count information. On the other hand, for route error messages, a node uses digital signatures to sign the whole message, and any neighbor of the node that receives such a route error message authenticates the signature.

Papadimitratos and Haas [39, 57] proposed the Secure Message Transmission (SMT) protocol, which is based on the notion of information dispersion. SMT assumes that there is another underlying protocol capable of discovering routes in the network (e.g., SRP [7]), although the routes may contain malicious nodes. SMT adds redundancy and partitions the information into fragments, while transmitting the fragments across multiple routes, so that even if some of the fragments are lost (i.e., those that are sent over the routes with malicious nodes), the remaining fragments suffice to reconstruct the original transmission. While SMP transmits data simultaneously over multiple routes, a modification of SMT, the Secure Single Path (SSP) protocol, transmits data over multiple routes in an alternative manner. The salient feature of these protocols is that they do not rely on trustworthiness of nodes in the network (with the exception, of course, of the source and the destination nodes). Indeed, the protocols can deliver highly reliable and low-delay communication even when a large fraction of the network nodes act maliciously. The protocols are, in particular, useful for reliable and secure real-time communications, when retransmissions may not be an option. As a point of reference, SMT can deliver 93% of messages without retransmissions, even when 50% of the nodes randomly drop packets. Figure 11 depicts an example of the SMT operation.

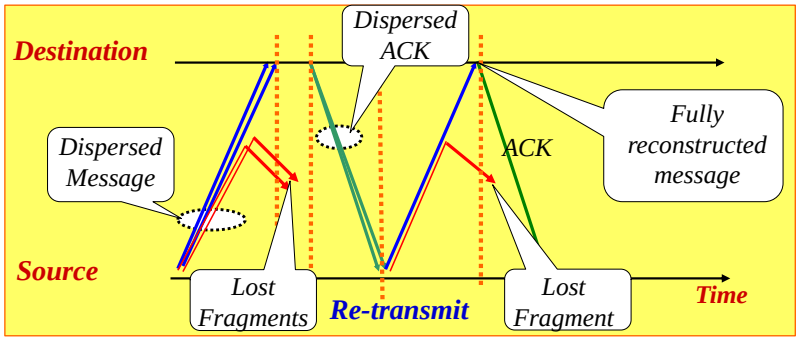

Figure 11: SMT transmission of a single message through dispersion

Ariadne [40], by Hu, Perrig, and Johnson, is an on-demand secure routing protocol for ad hoc networks. Ariadne's goal is to prevent attacks by tampering with uncompromised routes (i.e., routes formed by uncompromised nodes), and also prevent many types of DoS attacks. The operation of the Ariadne protocol is based on the target node authenticating the Route Requests. This is accomplished by the initiator including a

MAC, which is calculated over the unique data in the Route Request, and using key K_{sd}. Ariadne uses three alternative mechanisms for route data authentication, which are: TESLA protocol, Digital signatures, and standard MACS. Additionally, per-hop hashing technique is used to confirm that there is no missing node in the list of nodes in the request. The design of the protocol is based on a reactive routing protocol DSR [48] and a broadcast authentication protocol TESLA [58, 59]. The operation of DSR is divided into *Route Discovery* and *Route Maintenance*. In *Route Discovery*, the source node *S* broadcasts a ROUTE_REQUEST message containing the identifier for the destination node *D* (which is referred to as the "target"). There are two situations in which discarding of a ROUTE_REQUEST occurs. One such a situation is when the node's address is already listed in the route's record. Discarding the request avoids a request propagating around a loop. The other such a situation involves discarding the request upon determination that the host has recently seen a copy of the same request, one carrying the same initiator address and the same request id. This guarantees that a later copy of the request that arrived at this node by a different route is removed. If the request is not discarded, the node appends its own identifier to the ROUTE_REQUEST message and re-broadcasts the message. When ROUTE_REQUEST reaches the target *D*, *D* replies with a ROUTE_REPLY message, which contains the routing information, back to the source node *S*. Node *S* then uses the route in ROUTE_REPLY message to forward data to the node *D*. *Route Maintenance* is a mechanism used to detect broken links on an established route. If any of the intermediate hop transmission along the path fails, the node unable to make the next hop transmission returns a ROUTE_ERROR message back to *S*, and *S* repeats the *Route Discovery* phase. TESLA uses a secret key to generate message authentication code (MAC) for messages to ensure broadcast authentication. The secret key should be kept secret by the message originator, so that no other node can forge the MAC, however, the receiving nodes need the secret key for verification. Instead of using a computationally expensive asymmetric cryptography scheme such as RSA [19], TESLA achieves this asymmetry from loose time synchronization and delayed key disclosure. In TESLA, the sender chooses a random initial key K_N, and generates a one-way key chain by repeatedly computing a one-way hash function *H* on its starting value: $K_i = H(K_{i+1}) = H^{N-i}(K_N)$. To compute any previous key K_j from a key K_i in which $j < i$, a node can compute $K_j = H^{i-j}(K_i)$ Then, at time slot t_i, the key K_i is used to generate the MAC. At the next time slot t_{i+1}, K_i is published, so that the receiving nodes can verify the MAC using K_i. If the source node has additional broadcast messages to send, K_{i+1} is used to generate the MAC for the new messages at time t_{i+1}. Similar to DSR, Ariadne also has a *Route Discovery* phase and a *Route Maintenance* phase. To support secure routing, the ROUTE_REQUST, ROUTE_REPLY, and ROUTE_ERROR messages are all authenticated using the TESLA scheme described above.

Marti et al. in [38] proposed a reputation-based secure routing for ad hoc networks. In reputation-based routing, the next-hop of a path is chosen based on reliability of links and reputation of nodes. Marti et al. used a watchdog that identifies misbehaving nodes and a path-rating scheme that helps routing protocols to avoid these nodes; Watchdog and Pathrater are the two mechanisms used to detect and mitigate routing misbehavior. These mechanisms are implemented on top of source routing protocols. Detection of misbehaving nodes is done by the Watchdog by keeping a buffer of packets that were

recently sent. The Watchdog then attempts to verify whether a packet has, indeed, been forwarded by the next node by overhearing the transmissions of the neighboring nodes. The Watchdog removes the packet from its buffer when it determines that the packet has been forwarded by the next node. If after a timeout the packet is still in the buffer, a failure count of the node that was responsible for forwarding on the packets is incremented by the Watchdog. If the count surpasses a particular threshold, the node is considered a misbehaving node. The Pathrater is run by every network node. The most likely reliable route is chosen by taking into the consideration the knowledge of misbehaving nodes and the data about links' reliability.

The algorithm of the Pathrater assigns ratings to nodes in five steps: firstly, when the Pathrater becomes aware of a node in the network, the Pathrater assigns the node the rating of 0.5 (every node assigns itself the value of 1.0.) Secondly, at periodic time intervals (of 200ms), the Pathrater increments the ratings of nodes on all actively used paths by the value of 0.01, with the maximum rating value being 0.8. Thirdly, when the Pathrater detect a misbehaving during packet forwarding, it decrements a misbehaving node's rating by 0.05. Fourthly, negative path values suggest that there is one or more suspected misbehaving nodes on the path. Of course, the goal is to choose the path with the highest ratings.

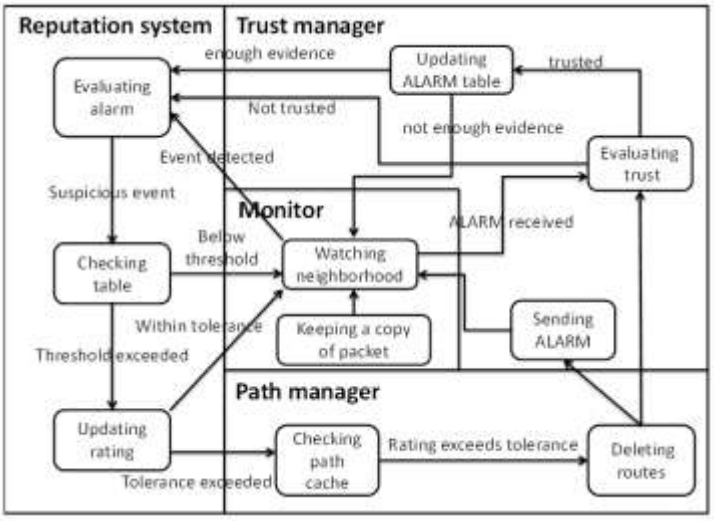

Figure 12: The relationship between monitor, reputation system, path manager, and trust manager. (Based on [41])

The Grudger Protocol by Buchegger and Boudec in [41] is also a reputation-based secure routing for ad hoc networks. Detecting and isolating misbehaving nodes becomes possible by utilizing the Grudger Protocol. Trust relationships and routing decisions are based on behavior of other nodes, which is gathered through experience, observation, or reports. It is intended to be implemented and run on top of any existing ad hoc routing protocols, such as DSR or AODV. Each node of the Grudger protocol consists of four components: monitor, reputation system, path manager, and trust manager. The monitor detects deviance by watching its neighborhood. This is accomplished by listening to the transmissions of the next node to verify that it forwards the packet. As a result,

nonconformities can be detected. The trust manager plays an important role in three aspects: firstly, using trust function it calculates the trust levels of nodes and manages trust levels in a trust table; secondly, the incoming ALARM messages are filtered based on the trust level of the reporting nodes and maintaining information about received alarms in an alarm table; thirdly, forwarding the ALARM messages according to the friends list. The reputation system is in charge of maintaining a table of the ratings of the nodes. A rating is determined based on a function that includes the node's own experience, the observations, and the reported experience. The path manager re-ranks paths based on a security metric, deletes paths which contain malicious nodes, ignores route requests generated by malicious node, and ignores requests for a route which contains a malicious node in the source route (while alerting the source node).

Figure 12 describes the relationship between the four components of the Grudger Protocol (the monitor, the reputation system, the path manager, and the trust manager).The operation is explained as follows in four steps. Step 1: When the monitor detected a suspicious event, such information is passed on to its reputation system. Step 2: The reputation system determines whether the event happened more often than some predefined threshold, and if so, the rating of the node that caused the event is adjusted by the reputation system. Step 3: If the resulting rating of the node is too high, then the path manager removes all the routes that contain this node from the cache of the paths. The trust manager then sends out an ALARM message. Step 4: Upon receipt of such an ALARM message by a monitor component from a node that is (at least) partially trusted, the monitor passes such a received ALARM message on to the trust manager, and the ALARM table in the trust manager is updated. Depending on the level of evidence, the information about the node reported in the ALAM will be passed on to the reputation system.

Despite the effectiveness of those secure routing protocols for ad hoc networks, they may not always be directly applied to WSNs. This is because WSNs usually have a directional data flow, from the data collector nodes towards the sink nodes; whereas in ad hoc networks, data flows are more uniform among nodes. Routing protocols designed for ad hoc networks do not take this directional data flow characteristic into considerations. Therefore, depending on the situation WSNs may need their own versions of such secure routing protocols.

3.2 Multipath Routing

Multipath routing protocols take advantage of the redundancy of sensor nodes in the network. They are robust against limited number of compromised nodes, at the expenses of larger communication overhead. INSENS (Intrusion-tolerant Routing Protocol for Wireless Sensor Networks), proposed by Deng et al. in [60], is an example of a multipath routing protocol. The main goal of INSENS is to support operation in spite of the harm caused by an intruder who was able to compromised sensor nodes with the intention to inject, modify, or block packets. INSENS assumes that after the initial deployment, sensor nodes can have only bounded mobility. INSENS is effective against DoS attacks and false routing information spreading. In this scheme, each node shares a secret key only with the sink node and not with other nodes. To defend against DoS attacks,

broadcast is only permitted by the sink node. To defend against false routing information spreading, initially the sink node computes a multi-hop, multi-path data forwarding tree in three rounds. In round 1, the sink node broadcasts route request message to all sensor nodes. In round 2, the sensor nodes reply back to the sink node with their local topological information. In round 3, the sink node computes a routing table, and then securely unicasts it to each sensor node in a breadth-first manner. This forms a data forwarding tree rooted at the sink node. Data forwarding then proceeds according to this data forwarding tree. Multipath routing in INSENS enhances intrusion tolerance, so that even if an intruder compromises a node or a path, alternate forwarding paths still exist on the data forwarding tree. Bidirectional verification is used to protect against the rushing attack. Nodes joining and leaving a network are supervised by secure maintenance mechanisms.

In Figure 13, multiple routes are derived between each source and destination. The intent is that these paths should be as independent as possible; i.e., that the paths share in common minimum number of nodes and links. In the best case, only the source node and the destination node are common between two paths. In fact, the second path should exclude the nodes on the first path (area S1 in Figure 13), their neighbors (area S2), and the neighbors of their neighbors (area S3). One or more intruders along some paths can jeopardize the delivery of some of the copies of a message. However, as long as there is at least one path that is not affected by an intruder, the destination will receive a correct copy of the message.

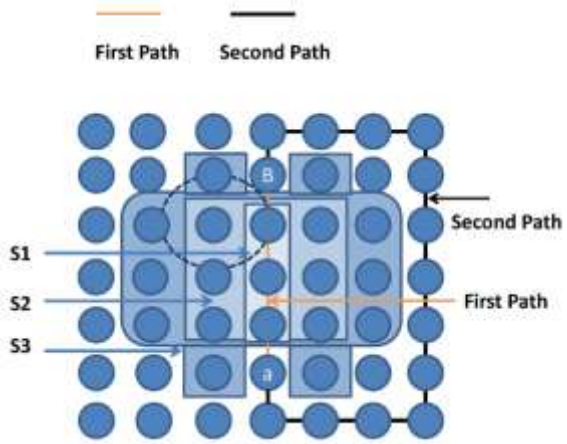

Figure 13: Selection of paths in the multipath routing policy. (From [60])

3.3 Secure Routing for Cluster or Hierarchical Sensor Networks

Many WSN architectures are "cluster-based". In such architectures, each cluster has a cluster head and many subordinates, and the cluster head is very close (one-hop or only few-hops away) from all subordinates in its cluster. Subordinates collect data and send the data to the cluster head, and then the cluster head determines routing path and transmits aggregated data. LEACH in [61], proposed by Heinzelman et al., is one of the first cluster-based routing protocols that significantly reduces energy consumption. LEACH is a self-organizing, adaptive clustering protocol that uses randomization to distribute the energy load evenly among the sensor nodes. After sensor node deployment, nodes cluster themselves and elect one cluster head for each cluster. Since cluster heads

typically perform more intensive processing, they are more prone to faster battery drainage and reduced lifetime. To reduce this problem, LEACH randomly rotates the cluster-head position. Furthermore, to reduce energy and to enhance system lifetime, the transmissions to the cluster head are compressed using local data fusion. The cluster head selection process is done probabilistically: each sensor node elects itself to be a local cluster head with certain probability. A cluster head broadcasts its status (e.g., remaining energy, location information, etc.) to other sensors. The non-cluster head nodes then join a cluster by choosing the cluster head that requires the minimum communication energy. After all the nodes are arranged into clusters, every custer-head generates a schedule to be used by the nodes that belong to the cluster-head. The energy dissipated in the sensor can be minimized by turning off the radios of nodes, when the nodes are not transmitting. The LEACH protocol also equalizes the energy used by the nodes, so that nodes are depleted of energy at about equal rate, thus allowing maintaining a more uniform coverage of the environment.

Operation of the LEACH algorithm is based on rounds, and the algorithm comprises the following phases: (a) the advertisement phase, (b) the cluster set-up phase, (c) the schedule creation phase, and (d) the data transmission phase.

There are two steps in the advertisement phase; the cluster-heads are chosen in the first step by having each node, n, select a random number between 0 and 1. If this number is less than the threshold $T(n)$, then the node serves as a cluster-head in the current round. The threshold function is set as follows:

$$T(n) = \begin{cases} \dfrac{P}{1 - P \cdot (r \bmod P^{-1})} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases}$$

where P is the intended percentage of cluster heads, r is the number of the current round, and G represents the set of nodes that have not served as cluster-heads in the last $1/P$ rounds. The second step in the advertisement phase consists of forming the clusters: transmitting with the same power, the cluster-heads transmit their advertisement using the CSMA (Carrier Sense Multiple Access) protocol. Each non-cluster-head node selects its cluster-head (and, thus, the cluster) for this round based on the measured signal strength of the received advertisement transmissions. During the cluster set-up phase, a non-cluster-head node transmits its selection to the cluster-head and, thus, becomes a member of the cluster. The cluster-head node generates a TDMA schedule, which is based on the number of the nodes in its cluster, and which indicates to each node when a node can transmit. In the data transmission phase, the non-cluster nodes transmit to the cluster head based on the TDMA schedule. When all the data from the non-cluster nodes have been received by the cluster head, the cluster head compresses the data into a single signal, which the cluster head then transmits to the base station. To reduce the interference between the transmissions of the nodes in different clusters which are in close proximity one to another, the clusters use different CDMA (Code Division Multiple Access) codes to communicate. The cluster head send the information about the choice of a particular spreading code to the nodes in its cluster. Using the particular spreading code, the cluster head can then extract

the information sent by its nodes, while reducing the interference caused by transmission of nodes in other clusters.

Despite the energy-efficiency characteristic of LEACH, Karlof et al. in [43] pointed out that such cluster-based protocols are susceptible to the Sybil attack, in which a compromised node can claim multiple identities and advertises itself as multiple cluster heads. Tubaishat et al. proposed in [62] a cluster-based routing protocol, the Secure Routing Protocol for Sensor Networks (SRPSN). The goal of the SRPSN protocol is to protect the data packet in the sensor networks from different types of attacks. It uses a group key management scheme, which contains group communication policies, group membership requirements and an algorithm for generating a distributed group key for secure communication. This secure routing protocol stores the routing table in a cache. The protocol uses hierarchical architecture and highly efficient symmetric cryptographic operations. In the group key management scheme, the key computation starts from the initiator node by using its partial key. The group key is computed by a leader using the partial keys which are contributed by every sensor node in a group; i.e., one can consider this as a "bottom up approach," as the accumulation of the partial keys is done from a leaf nodes up to the parent nodes. A modified multiparty Diffie-Hellman protocol [63] is used for computing group key, while the updating of group keys is done using the concept of key trees. The routing information messages (e.g., route request and route reply messages) are then encrypted using the group key.

Cluster-based secure routing protocols are also used in wireless sensor and actuator networks (WSANs), which is a special type of WSNs that consists of both, low-power sensor nodes that form the traditional WSN and high-power actuator nodes. Therefore, while WSNs are mostly concerned only with sensors-to-sensors communications, WSANs have to consider four types of communications: sensors-to-sensors, sensors-to-actuators, actuators-to-sensors, and actuators-to-actuators. Furthermore, the natures of the four types of communications are different. For example, sensors-to-sensors communication is usually many-to-one (sensors to the sink) or many-to-many (sensors to sinks), whereas actuators-to-actuators communication is usually peer-to-peer. Huet al., in [64] proposed a secure routing protocol based on WSAN's hierarchical network architecture. A scalable and energy-efficient routing architecture, referred to as Ripple-zone (RZ), is employed to implement WSAN security. A multiple-key management scheme, together with the Ripple-zone routing architecture, improves the security of in-network processing, for example, of the data aggregation operation. The scheme uses a Member Recognition Protocol (MRP) to allow actuators and sensors to self-organize themselves into separate domains, with each actuator as the domain center. As shown in Figure 14, within each domain, sensor nodes are grouped into ripple zones around the domain center actuator, such that nodes in a ripple zone all have the same number of hops to the actuator. Within each ripple zone, sensor nodes are further clustered, and each cluster elects a sensor node as the cluster head or "master". A "master" is responsible to accumulate data from the sensors in its zone. The "master" then transmits the data to the "master" in the next "ripple," which is located closer to the actuator. Each node (sensor or actuator) shares a global key and a pairwise key with the sink node, which are updated periodically. In the high-level (among actuators), two types of keys exist: session key (SK), which is used to secure data packet transmission, and a backbone key (BK), which

is used to secure control packets that include session key re-keying information. Figure 15 shows the relationship between these two keys. To protect against attacks, the session keys (SKs) are periodically re-keyed, while the refreshing of the backbone key (BK) is event-triggered, based on events such as actuator insertion, node death, or node compromise. A chain of session keys is generated at the sink node by continuously applying a known one-way hash function, and the key chain is sent to the actuators. An actuator keeps a buffer to store the key chain in order to tolerate multiple key losses. To support different security levels for different types of messages, multiple types of keys are introduced in the low level: Master-to-Actuator Key (MAK), Inter-Master Pairwise Key (MPK), Sensor-to-master Pair-wise Key (SPK), Zone Key (ZK), and Ripple Key (RK). A MAK is shared between each master and its domain actuator and is used for direct master-to-actuator secure communication. MPK is used occasionally to establish secure channels between two masters that belong to two actuator domains. SPK is shared between a master and each of the sensors in its zone. ZK is used for data aggregation and also for propagation of a query message to the whole cluster and is shared among all sensors in the same cluster. RK is used to achieve hop-to-hop security in an actuator domain. This multi-key management scheme allows for the establishment of a secure routing protocol based on ripple-zone, in which messages are routed in a hop-by-hop manner across ripple zones.

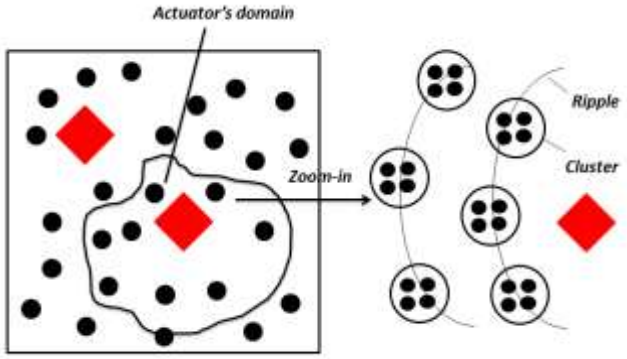

Figure 14: Ripple-zone-based WSAN routing. (From [64])

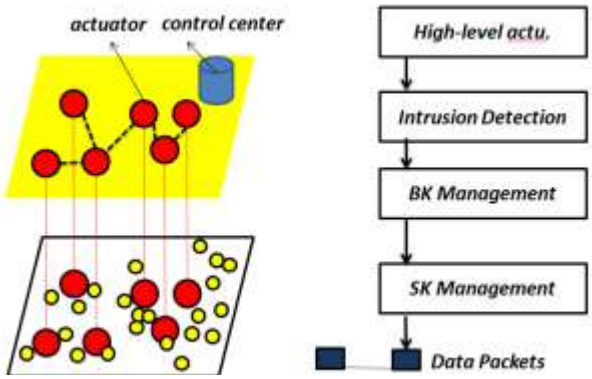

Figure 15: Backbone Key (BK) and Session Key (SK). (From [64])

3.4 Broadcast Authentication

Broadcast is a fundamental operation in many networks, and it is an essential component in many routing protocols. Perrig et al. in [65] proposed μTESLA, an authenticated broadcast protocol for the SPINS (Security Protocols for Sensor Networks). In general, authentication operation based on asymmetric cryptography is too computationally intensive for WSN nodes. μTESLA overcomes this problem through the concept of delaying the disclosure of symmetric keys. To send an authenticated packet, the sink node computes a MAC (message authentication code) on the packet with a key that is secret at that point in time. Upon receiving the packet, the node temporarily stores the packet in a buffer and waits for the key disclosure from the sink node. At the time of key disclosure, the sink node broadcasts the verification key to all the receivers. When a node receives the disclosed key, it can verify the key. If the key is correct, the node can use it to authenticate the packet. Each MAC key K_i is a key of a key chain, generated by a public one-way function F, such that $K_i = F(K_{i+1})$, where the subscript denotes the time interval.

4. Secure Localization Schemes

Depending on the application, localization can be an essential service in a WSN. For example, a location-based routing protocol, GPSR (geographic routing protocol) proposed by Karp et al. in [66], relies on accurate position of sensor nodes to perform routing. Localization is a well-studied topic, but almost all localization systems operate in a non-adversarial setting [67]. Secure localization only recently emerged as an active area of research. Secure localization schemes can be categorized into two groups—beacon-based and non-beacon-based.

4.1 Beacon-Based Schemes

In the beacon-based schemes, some nodes in the WSN (referred to as *beacon nodes*) are equipped with GPS hardware. These beacon nodes can correctly identify their own location via GPS signals. The beacon nodes can then help the non-beacon nodes to obtain their location information. The localization schemes can be categorized into two types of schemes: "range dependent" and "range-independent". In the range-dependent schemes, the calculation of a node's location is based on the estimates of distances and angles to reference points, where the locations (coordinates) of the reference points are known. Such estimates are usually obtained by one of the following ways: received signal strength, Time of Arrival (ToA), Time Difference of Arrival (TDoA), and Angle of Arrival (AoA) [67].

On the other hand, the range-independent localization schemes do not rely on the nodes performing time, angle, or power measurements. For example, Lazos et al. in [68] proposed a range-independent localization algorithm called SeRLoc that is beacon-based. SeRLoc is a distributed algorithm based on a two-tier network architecture that allows sensors to passively determine their location without interacting with other sensors. There are two types of nodes: sensor nodes equipped with omnidirectional antennas, and locator

nodes equipped with multi-directional antennas and GPS. Locator nodes first obtain their accurate location via GPS, and then each locator transmits beacons with their individual coordinates and coverage areas. Each sensor node collects location information from all the locator nodes that it can receive and then, using this information, assembles a search area of its own location. After receiving enough beacons from different locators, the sensor estimates its location as the center of gravity of the overlapping region of the coverage areas. After analytically evaluating the probability of sensor displacement due to security threats in WSNs, such as the wormhole attack, the Sybil attack, and compromise of network entities, they showed that SeRLoc provides accurate location estimation even in the presence of these threats. See Figure 16 for additional details.

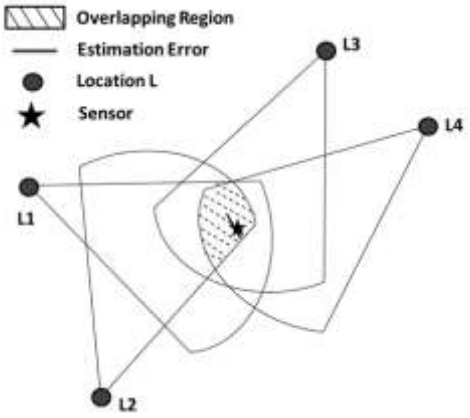

Figure 16: A sensor estimates its location as a Center of Gravity based on the beacons from locators L1, L2, L3, and L4. (Based on [68])

Li, et al. in [69] proposed a robust statistical method for secure localization using triangulation. In triangulation, a sensor node gathers a collection of {(*x*,*y*,*d*)} values, where *d* is an estimated distance from the sensor node to a beacon node at location (*x*,*y*). In the ideal case, these {(x,y,d)} values map out to a parabolic surface $d^2(x,y) = (x-x_0)^2 + (y-y_0)^2$. Thus, to estimate its location, the sensor node can simply solve for a Least Square problem from the gathered data set {(*x*,*y*,*d*)}. However, in the presence of adversaries, some of the (*x*,*y*,*d*) values can be outliers. Therefore, instead of using Least Square, the authors proposed to use Least Median Square [70] for achieving robustness in localization. Unlike Least Square, which minimizes the sum of the residue squares, Least Median Square minimizes the median of the residue squares. As a result, outliners have a much smaller effect on the optimization cost function, which makes the location estimation more robust. Contamination ratio (ϵ) is the fraction of the samples that are outliers and the noise level is assumed to be σ_n. Figure 17 shows the square root of mean square error (MSE) as a function of distance $d_a = \sqrt{(x_a - x_0)^2 + (y_a - y_0)^2}$ (measurement of the strength of the attack). In particular, the performances at two pairs of ϵ and σ_n values are presented in the figure; (a): (ϵ, σ_n)=(0.2, 20) and (b): (ϵ, σ_n)=(0.3, 15). The results demonstrate that the estimation error of ordinary LS increases with d_a, which is caused by the non-robustness of LS to outliers. On the other hand, the estimation error of LMS exhibits a different behavior; it first increases until reaching a maximum (which occurs at a critical value of d_a), then the

estimation error slightly decreases, and finally stabilizes. These results could be interpreted as saying that, if LMS is used for localization, the adversary does not gain by mounting a too powerful attack.

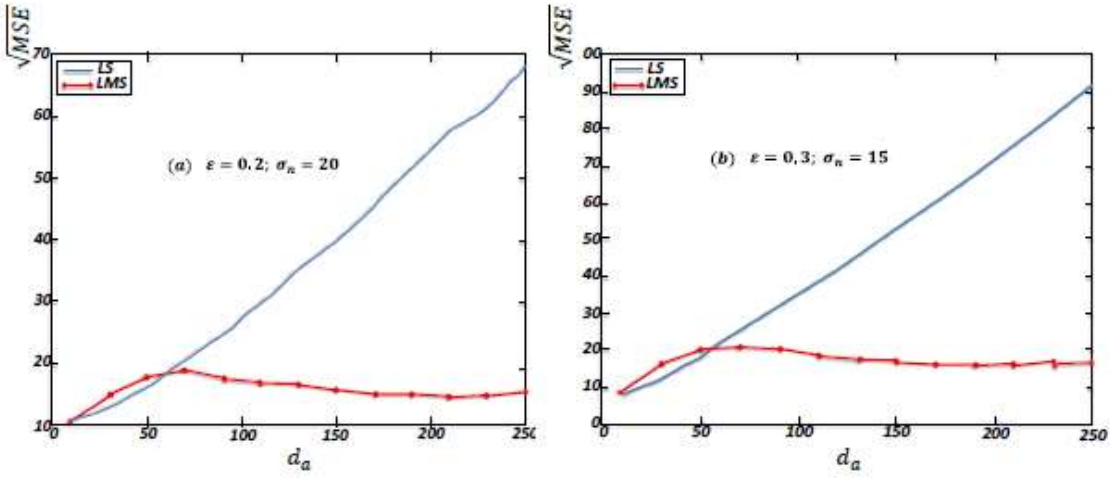

Figure 17: The performance comparison between LS and LMS for localization. (From [70])

4.2 Non-Beacon-Based Schemes

Since equipping sensor nodes with GPS hardware can be costly, in some practical environments beacon-based localization schemes may not be feasible. In non-beacon-based schemes, a node calculates the position of another node of interest by making an estimation based on the known locations of existing nodes. Non-beacon-based schemes are less accurate, but are also less expensive to implement than beacon-based schemes [67].

Fang et al. in [71] proposed a non-beacon-based localization scheme. The scheme is based on the following observation: in practice, it is quite common for sensor nodes to be deployed in groups. The locations of the groups (deployment points) are pre-determined prior to deployment and are stored in each sensor's memory. Sensors from the same group can be placed in locations which follow some *a priori* known spatial probability distribution, for example, a two-dimensional Gaussian distribution. With this prior deployment knowledge, sensors can estimate their locations by observing the group memberships of its neighbors. The scheme modeled the localization problem as a statistical estimation problem and used the Maximum Likelihood Estimation method to estimate the location.

5. Secure Data Aggregation

WSN applications that involve extensive amount of data processing typically do not have all the processing done at the central sink node, but instead some processing could be done by the network. Such in-network processing via data aggregation in large-scale sensor networks has been shown to improve scalability, eliminate information redundancy, and increase the lifetime of the network, but the drawback is that data aggregation renders the security problem more difficult [72].

In a large-scale data processing network, sensor nodes can be classified into two groups. Most of the nodes are data collector nodes that are only responsible for collecting sensor

measurements. The other nodes are data aggregator nodes that perform aggregation functions upon receiving data from collector nodes. The massive data processing performed by the collector-aggregator architecture can significantly reduce the communication overhead in the network. However, from a security perspective, there are two types of potential threats: the first one is that aggregators can receive false data from collectors; the second one is that the sink node receives false data from compromised aggregators [10]. Secure data aggregation schemes are developed to overcome these two threats. These schemes can be classified into plaintext-based schemes and cipher-based schemes.

5.1 Plaintext-Based Schemes

In the plaintext-based secure data aggregation schemes, intermediate nodes in the path can read the data in transit. Hu, et al. in [73] proposed such an example. In this scheme, each node A is initialized before deployment with a symmetric secret key, K_{AS}, shared with the sink node. Time is slotted, so that as time progresses, a sequence of temporary encryption keys will be generated for node A. For example, in time slot i, the temporary encryption key for A would be $K_{Ai} = E(K_{AS}, i)$. After each time slot i, the temporary encryption key K_{Ai} will be revealed to all sensor nodes. The data aggregation proceeds as follows. Consider the following sequence of connected nodes: $A \rightarrow B \rightarrow C \rightarrow S$, as in Figure 18. At time slot i, node A transmits its reading R_A, identifier A, and a message authentication code $MAC(K_{Ai}, R_A)$ to its next hop node B. Node A will hold the data until time slot $i + 1$, when K_{Ai} is revealed. This same sequence of operations is done at node B (i.e., at the time of slot i, B sends $\{R_B, B, MAC(K_{Bi}, R_B)\}$ to C). At the time of the slot $i + 1$, K_{Ai} and K_{Bi} are revealed to all the nodes. Therefore node B can verify the integrity of R_A, and if R_A is verified, then B forwards A's message $\{R_A, A, MAC(K_{Ai}, R_A)\}$ to C. Similarly, C can verify R_A's and R_B's integrity using K_{Ai} and K_{Bi}, respectively. If the verification test at C passed, C can perform aggregation over R_A and R_B. This data aggregation scheme is therefore a delayed aggregation—aggregation is performed not at the immediate next hop, but at a later hop. As a result, the intermediate node has to forward both its own data and the received data to the last node, and therefore an additional transmission cost is incurred. However, delayed aggregation benefits data integrity—an adversary who obtains key material from a compromised node cannot tamper with many sensor readings.

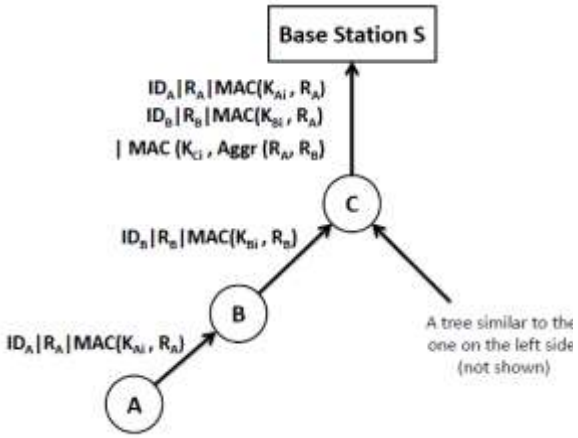

Figure 18: Example Sensor Network. (Based on [73])

5.2 Cipher-Based Schemes

In cipher-based secure data aggregation schemes, intermediate nodes on the path cannot read the data in transit. One implementation of such a scheme is Concealed Data Aggregation (CDA), proposed by Girao, et al. in [74]. CDA is based on a concept called *privacy homomorphism* (PH) proposed by Domingo-Ferrer in [75]. PH is a particular encryption transformation with additive and multiplicative homomorphic properties, so that direct computation over encrypted data is possible. Suppose Q and R are two rings, where "$+$" and "$*$" are the corresponding addition and multiplication operations for both rings. Let K denote the set of keys, E to be an encryption function ($E: K \times Q \to R$), and D to be a corresponding decryption function ($D: K \times R \to Q$). Then PH ensures that, for all $a, b \in$ Q and $k \in$ K, we have $a + b = D_k\big(E_k(a) + E_k(b)\big)$ (i.e., homomorphic addition), and $a * b = D_k(E_k(a) * E_k(b))$ (i.e., homomorphic multiplication). CDA uses PH to encrypt aggregated data along the path. The additive and multiplicative homomorphism properties allow processing data while the data is encrypted, without the necessity to decrypt the data at each intermediate node. This allows preservation of data confidentiality and integrity while the data is routed within the network.

6. Conclusion

In this chapter, we discussed several important aspects of WSN security, including cryptography schemes, key management schemes, secure routing protocols, secure localization, and secure data aggregation.

Cryptography schemes are classified into public key cryptography and symmetric key cryptography. Public key cryptography schemes are more computationally demanding, but require less care in key distribution and management. Due to limited resources at the sensor nodes, public key cryptography schemes are often considered infeasible for WSNs, although recent results showed that some public key cryptography schemes can be implemented in WSNs by choosing appropriate algorithms, parameters, etc. However, achieving energy-efficient public key cryptography schemes still need further research. For symmetric key cryptography schemes, efficient key management schemes need to be designed.

We discussed four categories of key management schemes—key pre-distribution schemes, hybrid cryptography schemes, key infection schemes, and key management in hierarchical networks. Although key management has been an active research area in the past decade, there are still certain open problems in this area. Current key management schemes are mostly concerned with static WSNs, and key management schemes for mobile WSNs still lack appropriate solutions. Most key management schemes require trustworthy sink nodes, which may not be a valid assumption in many applications; therefore new schemes need to be designed to secure the sink node.

Currently there are many secure routing algorithms for WSNs, and many of them are derivatives of secure ad hoc network routing algorithms. We reviewed several ad hoc network routing protocols, then surveyed two categories of secure routing protocols

specifically designed for WSNs—multipath routing and cluster-based routing. Although many secure routing algorithms can prevent or detect node compromise to some extent, there is still a window of vulnerability in which a compromised node can go unnoticed and false routing information may be spread. Designing secure routing protocols to minimize this window of vulnerability is another important research area. Yet another consideration for future secure routing research is to expand the evaluation metrics. Current evaluations of secure routing are mostly focused on security metrics; other metrics such as QoS need to be considered in addition to security.

Secure localization is divided into two categories: beacon-based and non-beacon-based schemes. However, both types of schemes are only suitable for static WSNs. Mobile WSNs secure localization still need further investigation.

Secure data aggregation schemes include plaintext-based and cipher-based schemes. Data aggregation schemes usually assume aggregators as more powerful sensor nodes than data collector sensor nodes. Therefore it is desirable to design secure data aggregation schemes that can be applied in a homogeneous WSN, where all the sensor nodes have equal capabilities. Another potential research direction in secure data aggregation is to investigate the tradeoffs between security and energy efficiency gains.

A somewhat newer topic related to WSN that has not been covered in this chapter is that of security of *Internet of Things* (*IoT*) networks. The reader is referred to references [76]-[81].

Acknowledgements

This work was sponsored in part by the NSF grants numbers ANI-0329905, CNS-0626751, CNS-1040689, ECCS-1308208, CNS-1352880, and by the AFOSR contract number FA9550-09-1-0121/Z806001.

References

[1] M. Bishop, "Computer Security: Art and Science,"Addison-Wesley, 2003

[2] G. Stoneburner, C. Hayden, and A. Feringa, "Engineering principles for information technology security," NIST Special Publication 800-27, Revision A, June 2004.

[3] M. Healy, et al., "Wireless Sensor Hardware: A Review," IEEE Sensors, 2008

[4] J. P. Walters, et al., "Wireless Sensor Network Security: A Survey," Security in Distributed, Grid, and Pervasive Computing. Ed. Y. Xiao, CRC Press, 2006.

[5] E. Shi, et al., "Designing Secure Sensor Networks," IEEE Communication Magazine, 2004.

[6] A. Wood, et al., "Denial of Service in Sensor Networks," IEEE Computer, 2002.

[7] P. Papadimitratos and Z. J. Haas, "Secure routing for mobile ad hoc networks," in *Proc. SCS CNDS*, San Antonio, TX, Jan. 27–31, 2002, pp.193–204.

[8] J. Jeong, G.Y Lee, and Z.J. Haas, "Prevention of Black-Hole Attack Using One-Way Hash Chain Scheme in Ad Hoc Networks," *International Conference on Information Networking*, Estoril, Portugal, January 22-25, 2007.

[9] L. Zhou and Z.J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, 1999, pp. 24-30.

[10] X. Chen, et al., "Sensor Network Security: A Survey". IEEE Commun. Surveys & Tutorials, 2009

[11] S. A. Camtepe, et al., "Key distribution mechanisms for wireless sensor networks: A survey," Computer Science Department at RPI Tech, Rep. TR-05-07, 2005.

[12] P. Ganesan, et al., "Analyzing and modeling encryption overhead for sensor network nodes," in Proc. *2nd ACM International Conf. Wireless Sensor Networks Applications*, 2003, pp. 151-159.

[13] G. Gaubatz, et al., "State of the art in ultra-low power public-key cryptography for wireless sensor networks," in *Proc. 3rd IEEE International Conf. Pervasive Computing Commun. Workshops*, 2005, pp. 146-150.

[14] M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," MIT Laboratory for Computer Science, January 1979.

[15] J. Hoffstein, J. Pipher, J. Silverman, "NTRU: A Ring Based Public Key Cryptosystem," in Algorithmic Number Theory (ANTS III), Portland, OR, June 1998.

[16] N. Koblitz, "Elliptic curve cryptosystem," *Mathematics of Computation* 48 (177): 203-209. JSTOR 2007884.

[17] V. Miller, "Use of elliptic curves in cryptography," *CRYPTO* 85: 417-426.

[18] A. Wander, et al., "Energy analysis for public-key cryptography for wireless sensor networks," In *IEEE PerCom'05*, Pisa, Italy, Mar. 2005.

[19] R.L. Rivest, A. Shamir and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* 21(2) (1978) 120-126.

[20] F. Koushanfar, M. Potkonjak, Watermarking Techniques for Sensor Networks: Foundation and Applications, in *Security in Sensor Networks*, ed. by Y. Xiao (Auerbach Publications, Taylor & Francis Group, 2006)

[21] J. Jeong and Z.J. Haas, "Predeployed Secure Key Distribution Mechanism in Sensor Networks: Current State-of-the-Art and a New Approach Using Time Information," *IEEE Wireless Communication*, August 2008, pp. 42-51

[22] L. Eschenauer, et al., "A key-management scheme for distributed sensor networks," in *Proc. Conf. Computer Commun. Security*, 2002, pp. 41-47.

[23] H. Chan, et al., "Random key predistribution schemes for sensor networks," in *Proc.IEEE Symposium Security Privacy*, 2003, pp. 197-203.

[24] S. Zhu and W. Zhang, "Group Key Management in Sensor Networks," Book Chapter from *Security in Sensor Networks*, edited by Y. Xiao. Auerbach Publications. 2007.

[25] S. Zhu, et al., "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proc. 10th ACM Conf. Computer Commun. Security*, 2003 pp. 62-72.

[26] R. Blom, "An Optimal Class of Symmetric Key Generation Systems, Advances in Cryptology," Proc. of EUROCRYPT84, LNCS, Vol. 209, pp. 335-338, 1984.

[27] Q. Huang, et al. "Fast authenticated key establishment protocols for self- organizing

sensor networks," in *Proc. 2nd ACM International Conf. Wireless Sensor Networks Applications*, 2003, pp. 141-150.

[28] R. Anderson, et al., "Key infection: Smart trust for smart dust," in *Proc. 12th IEEE International Conf. Network Protocols (ICNP)*, 2004.

[29] G. Jolly, et al., "A low-energy key management protocol for wireless sensor networks," in *Proc. 8th* nternational Symposium Computers Commun. (ISCC), 2003, vol. 1, pp. 335-340.

[30] M. Chorzempa, et al., "SECK: Survivable and efficient clustered keying for wireless sensor networks," in Proc. IEEE Workshop on Information Assurance in Wireless Sensor Networks, pp. 453-458. Phoenix AZ, April 2005.

[31] J. Newsome, et al., "The Sybil Attack in Sensor Networks: Analysis & Defenses", in *Proc. 3rd International Symposium on Information Processing in Sensor Networks*, 2004, pp. 259-268.

[32] S. Ratnasamy, et al., "GHT: a geographic hash table for data-centric storage," In *WSNA 2002*, Sept.

[33] Y. Hu, et al., "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," in *Proc. IEEE INFOCOM*, 2003.

[34] C. Castelluccia, et al., "On the difficulty of software-based attestation of embedded devices," in *CCS'09*, November 9-13, 2009

[35] A. Seshadri, et al., "SWATT: Software-based attestation for embedded devices," in *Proc. IEEE Symposium Security Privacy*, 2004, pp.272-282.

[36] A. Seshadri, et al., "SCUBA: Secure code update by attestation in sensor networks," in *Proc. 5th ACM Workshop Wireless Security*, 2006, pp. 85-94.

[37] M. G. Zapata, "Secure ad-hoc on-demand distance vector routing," Mobile Computing and Communications Review, Volume 6, Number 3, 2002.

[38] S. Marti, et al., "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc.6th Annual International Conference Mobile Computing Networking*, 2000, pp. 255-265.

[39] P. Papadimitratos and Z.J. Haas, "Securing Data Communication in Mobile Ad Hoc Networks," *IEEE Journal on Selected Issues in Communications (JSAC)*, special issue on "Security in Wireless Ad Hoc Networks," vol.24, no.2, February 2006, pp.343-356.

[40] Y. Hu and A. Perrig, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *ACM MOBICOM*, Sept. 2002, pp. 12-23.

[41] S. Buchegger and J. L. Boudec, "Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks," *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing, IEEE Computer Society, Canary Islands, Spain, 2002, pp. 403-410.*

[42] M.C. Wong, et al., "Security Issues in Ad Hoc Networks," Book Chapter from *Security in Sensor Networks*, edited by Y. Xiao. Auerbach Publications. 2007.

[43] C. Karlof, et al., "Secure routing in wireless sensor networks: Attacks and countermeasures," *Elsevier's AdHoc Networks Journal, Special Issue on SensorNetwork Applications and Protocols*, 1(2-3):293-315, September 2003.

[44] Y. Wang and Y. Tseng, "Attacks and Defenses of Routing Mechanisms in Ad Hoc and Sensor Networks," Book Chapter from *Security in Sensor Networks*, edited by Y. Xiao. Auerbach Publications.2007.

[45] C.E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," ACM Conference on Communications Architectures, Protocols and Applications, Vol. 1 (1994), pp. 234-244.

[46] B. Bellur and R.G. Ogier, "A reliable, efficient topology broadcast protocol for dynamic networks," IEEE INFOCOM, Vol. 1 (1999), pp.178-186.

[47] P. Jacquet, et al., "Optimized link state routing protocol for ad hoc networks," IEEE International Multi Topic Conference, Vol. 1 (2001), pp. 62-68.

[48] D.B. Johnson and D.A. Malts, "Dynamic source routing in ad hoc wireless networks, Mobile Computing," edited by T. Imielinski and H. Korth, Kluwer Academic Publishers, 1996, pp. 153-181.

[49] C.E. Perkins and E.M. Royer, "Ad-hoc on-demand distance vector routing," IEEE Workshop on Mobile Computing Systems and Applications, Vol. 1 (1999), pp. 90-100

[50] M.S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," Wireless Networks, Vol. 1 No. 1 (1995), pp. 61-81.

[51] C.K. Toh, "Associativity-based routing for ad hoc mobile networks," Wireless Personal Communications, Vol. 4 No. 2 (1997), pp. 103-139

[52] V.D. Park and M.S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," IEEE INFOCOM, Vol. 1 (1997), pp.1405-1413.

[53] Z.J. Haas and M.R. Pearlman, "The Performance of Query Control Schemes for the Zone Routing Protocol," *IEEE/ACM Transactions on Networking*, vol.9, no.4, August 2001, pp.427-438, DOI: 10.1109/90.944341

[54] G. Pei, M. Gerla, and T.W. Chen, "Fisheye state routing: a routing scheme for ad hoc wireless networks," IEEE International Conference on Communications, Vol. 1 (2000), pp. 18-22.

[55] P. Samar, M.R. Pearlman, and Z.J. Haas, "Independent Zone Routing: An Adaptive Hybrid Routing Framework for Ad Hoc Wireless Networks," *ACM/IEEE Transactions on Networking*, vol.12, no.4, August 2004, pp.595-608

[56] M.R. Pearlman and Z.J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," *IEEE Journal of Selected Areas in Communications*, vol. 17, no.8, August 1999, pp. 1395-1414, DOI: 10.1109/49.779922

[57] P. Papadimitratos and Z.J. Haas, "Secure Message Transmission in Mobile Ad Hoc Networks," Elsevier Ad Hoc Networks Journal, vol.1, no.1, July 2003, pp.193-20

[58] A. Perrig, R. Canetti, D. Song and J.D. Tygar, "Efficient and secure source authentication for multicast," in: *Proceeings of the Network and Distributed System Security Symposium, NDSS'01* pp.35-46.

[59] A. Perrig, R. Canetti, J.D. Tygar and . Song, "Efficient authentication and signing of multicast streams over lossy channels", in: *Proceedings of the IEEE Symposium on Security and Privacy* (May 2000) pp. 56-73.

[60] J. Deng, et al., "INSENS: Intrusion-tolerant routing in wireless sensor networks," *Computer Commun.*, vol. 29, pp. 216-230, 2006.

[61] W. R. Heinzelman et al., "Energy-efficient communication protocol for wireless microsensor networks," in *33rd Annual Hawaii International Conference on System Sciences*, 2000, pp. 3005-3014.

[62] M. Tubaishat, et al., "A secure hierarchical model for sensor network," *ACM SIGMOD Record*, vol. 33, pp. 7-13, 2004.

[63] W. Diffie and M. E. Hellman. "Privacy and Authentication: An Introduction to Cryptography," *Proceedings of the IEEE*, 67(3): 397-427, March 1979.

[64] F. Hu, et al., "Scalable Security in Wireless Sensor and Actuator Networks," Book Chapter from *Security in Sensor Networks*, edited by Y. Xiao. Auerbach Publications. 2007.

[65] A. Perrig, et al., "SPINS: Security protocols for sensor networks," *Springer Netherlands Wireless Networks*, vol. 8, pp. 521-534, 2002.

[66] B. Karp, et al., "GPSR: Greedy perimeter stateless routing for wireless networks," In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pp. 243-254, ACM Press, 2000.

[67] K. Ravichandran and K. M. Sivalingam, "Secure Localization in Sensor Networks," Book Chapter from *Security in Sensor Networks*, edited by Y. Xiao. Auerbach Publications. 2007.

[68] L. Lazos, et al., "SeRLoc: Robust Localization for Wireless Sensor Networks," In *Proc.3rd ACM Workshop Wireless Security*, 2004, pp. 21-30.

[69] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Proc. 4th International Symposium Information Processing in Sensor Networks*, 2005.

[70] P. Rousseeuw and A. Leroy, "Robust regression and outlier detection," Wiley-Interscience, Sept. 2003.

[71] L. Fang, et al., "A beacon-less location discovery scheme for wireless sensor networks," in *Proc. IEEE INFOCOM*, 2005.

[72] T. Dimitriou and I. Krontiris, "Secure In-Network Processing in Sensor Networks," Book Chapter from *Security in Sensor Networks*, edited by Y. Xiao. Auerbach Publications. 2007.

[73] L. Hu, et al., "Secure aggregation for wireless networks," in *Proc. Symposium Applications Internet Workshops*, 2003, pp. 384-391.

[74] J. Girao, et al., *CDA: Concealed data aggregation in wireless sensor networks*, in *Proc. ACM WiSe*, 2004.

[75] J. Domingo-Ferrer, "A provable secure additive and multiplicative privacy homomorphism," in *Proc. Information Security Conf.*, 2002, pp. 471-483.

[76] L. Atzori, et al., "The Internet of Things: A survey," Elsevier's Computer Networks, June 2010.

[77] R. Roman, et al., Integrating Wireless Sensor Networks and the Internet: A Security Analysis," Internet Research, 2009.

[78] R. Roman, et al., “Do Wireless Sensor Networks Need to be Completely Integrated into the Internet?,” Furture Internet of People, Things and Services (IoPTS) eco-Systems, Brussels, 2009.

[79] R. Hummen, et al., “A Security Protocol Adaptation Layer for the IP-based Internet of Things,” Interconnecting Smart Objects with the Internet Workshop, 2011.

[80] T. Zahariadis, et al., “Securing wireless sensor networks towards a trusted Internet of Things,” IoS Press, ISBN 978-1-60750-007-0, pp. 47-56.

[81] R. Roman, et al., “Key management systems for sensor networks in the context of the Internet of Things,” Elsevier’s Computers & Electrical Engineering, March 2011.