31

Security and Privacy Protection in Wireless Sensor Networks

Sasha Slijepcevic
University of California at Los Angeles

Jennifer L. Wong
University of California at Los Angeles

Miodrag Potkonjak
University of California at Los Angeles

31.1 Introduction

Security and privacy protection are of extreme importance for many of the proposed applications of wireless sensor networks (WSNs). The list of potential applications that require protection mechanisms includes early target tracking and monitoring on a battlefield; law enforcement applications; automotive telemetric applications; room occupation monitoring in office buildings; measuring temperature and pressure in oil pipelines [1]; and forest fire detection. All these applications have unlimited benefits and potential; however, if the sensor information is not protected properly, possible compromises in user information, the environment, and even physical actuators could result.

The primary driving impetus for the development of sensor networks has been military applications, where security requirements are at their highest [2]. Although a WSN deployed on a battlefield can offer a reliable assessment of battlefield conditions without risking lives, an inadequately protected network could become a powerful weapon for an enemy. Strong security requirements for such applications are often combined with an inhospitable and physically unprotected environment. For commercial applications of WSNs, the issue of privacy protection is as important as secure and reliable functioning of a network. The protection of personal physiological and psychological information is expected by any user. As the applications of WSNs become more complex and widespread, the ability to protect such systems from any unauthorized access will become increasingly important.

Sensor networks operate in a variety of physical environments and under varieties of constraints. The limited resources of sensor nodes require the development of customized system architectures for each particular WSN application so that the sensor node resources are efficiently used. Because security and privacy protection mechanisms require a significant amount of computational and storage resources,

such mechanisms must be tailored to the corresponding sensor system architectures and security threats specific to a given physical environment. Section 31.2 describes the unique properties of WSNs and the security challenges that they bring. Security implications and corresponding security solutions for two basic WSN system architectures, cell-based WSNs and ad hoc WSNs, are discussed in Section 31.3. The overview of the privacy protection solutions proposed for WSN, as well as the solutions originally developed for other environments but applicable in WSN, is given in Section 31.4. Section 31.5 summarizes and concludes the chapter.

31.2 Unique Security Challenges in Sensor Networks and Enabling Mechanisms

WSNs share several important properties with traditional wireless networks, most notably with mobile ad hoc networks. Both types of networks rely on wireless communication, ad hoc network deployment and setup, and constant changes in the network topology. Many security solutions proposed for wireless networks can be applied in WSNs; however, several unique characteristics of WSNs require new security mechanisms. In this section, four characteristics specific to WSNs and their resulting security challenges are discussed. Additionally, it presents work performed in the areas, system-level security, mobile code, and metering, which can be foundations for the development of security techniques in WSNs.

31.2.1 Security-Related Properties

Four properties that are specific for WSNs and require attention are hostile environment, limited resources, in-network processing, and application-specific architectures.

- *Hostile environment.* WSNs can be deployed in hostile environments such as battlefields. In these cases, the nodes cannot be protected from physical attacks. Security information potentially could be collected from compromised nodes. The development of tamper-proof nodes is one approach to security in hostile environments. However, as shown in Anderson and Kuhn [3], the development of such systems is far from simple and certainly not cheap in terms of computational and memory requirements. Because of the physical accessibility of sensor nodes, the security mechanisms for WSNs are specifically concerned with situations in which one or more nodes are compromised.
- *Limited resources.* Sensor network nodes are designed to be compact and therefore are limited by size, energy, computational power, and storage. The limited resources limit the types of security algorithms and protocols that can be implemented. Security solutions for WSNs operate in a solution space defined by the trade-off between resources spent on security and the achieved protection. Limited energy available to nodes allows for new types of attacks, such as a sleep deprivation torture attack [4].
- *In-network processing.* Communication between the nodes in a WSN consumes most of the available energy, much less than sensing and computation do. For that reason, WSNs perform localized processing [5] and data aggregation [6]. An optimal security architecture for this type of communication is one in which a group key is shared among the nodes in an immediate neighborhood. However, in an environment in which the nodes can be captured, the confidentiality offered by the shared symmetric keys is easily compromised.
- *Application-specific architectures.* As a result of the previously mentioned properties, WSN system architectures must be designed to be application specific. The flexibility of a general-purpose architecture is traded for the efficient utilization of the resources. Almost every aspect of a WSN can be adjusted to improve performance and optimize resource consumption in a network for a particular application. This allows a network designer to determine the importance of various security threats and adjust security mechanisms to these threats.

31.2.2 System-Level Security

Three types of cryptographic tools have been developed for practical security of real-life systems: firewalls, honeypots, and intrusion detection techniques. Each will be discussed briefly in order to illustrate the types of approaches that exist in the field. Although these techniques may not be well suited for WSNs as proposed, modifications of their notions may be excellent security approaches for WSN.

A firewall is a policy enforcement point (node) for a part of a network designed to restrict access from and to that subnetwork. Several classes of firewalls exist: packet filtering according to a particular set of rules; access to particular servers or ports; or application-level firewalls that protect by remembering the state of the network connection. Firewalls still face denial of service (DoS) attacks and they try to address them by filtering suspicious connections. Among the several limitations of firewalls is the fact that they do not protect the network from insider attacks and that filtering can only be done against already known attacks.

Honeypots are systems placed on networks specifically for the purpose of being attacked or compromised [7, 8]. Because they are not designed for true use, they exist only to detect and collect information about security attacks. Advantages of honeypots include low false positives; ability to capture unknown attacks; and ability to facilitate interaction with the attacker in order to gain better insights into actions and thinking. Intrusion detection techniques aim at recognizing statistical or pattern irregularities in the incoming or outgoing traffic. The most recent approach to detection of Internet attacks is probabilistic deduction of the IP traceback [9–12]. Finally, virtual private networks are logical extensions of private networks over insecure channels provided by the Internet.

31.2.3 Mobile Code

Once deployed, access to the nodes in a WSN for management and code updates poses security threats and drains resources. Despite difficulties, mechanisms that allow changes in application and system code on the nodes are necessary. One feasible solution for remote configuration and application code updates is network-wide deployment of mobile code. A legitimate mobile code is injected into the network through several nodes and then spread throughout the network [13]. This subsection surveys proposed code manipulation approaches (attacks) and techniques for secure execution of mobile code. Among mobile code intrusion techniques, four have been most popular: viruses, Trojan horses, buffer overflow, and covert communication channels.

A computer virus [14] can be defined as a small program that attaches to the host computer and co-opts its resources for the purpose of creating new copies of the virus. Detailed analysis of viruses and models of their proliferation can be found in Cohen [15] and Kephart and White [16]. Trojan horses [17, 18] disguise themselves as programs that appear to perform a function while actually performing another function. Buffer overflow has been by far the most common type of attack of computer security in the last decade [19]. These attacks use the functions of a privileged program in such a way that the attacker can take control of the program and corrupt the computer. This is commonly achieved by making suitable code available in the program address space and then inducing a program to jump to that space with suitable parameters. Recently, the first constraint-based analysis technique for automated detection of buffer overflow has been proposed [20].

Covert communication channels [21] arise from resource sharing in computer systems. For example, a process with high priority can pass information to a process with low priority by interfering or refraining from interfering with the timing of the process. The most popular and simplest is the timed Z-channel, in which the communication alphabet consists of time values [22]. Numerous generalization of the timed Z-channel have been proposed and analyzed [23–25].

Smaller mobile devices have created a strong impetus for the development of mobile code security techniques. At least three major approaches for mobile code security have emerged: code signing, sandboxes, and proof-carrying code. Code signing follows a typical client- and server-authenticated handshake protocol such as SSL or WTLS [26]. Recently, sandboxing has attracted a great deal of attention [27] as

a security paradigm; Brigner [28] presented a 3-MB Java applet that implements a sandbox. In addition, Sekar and Uppuluri developed a security layer that includes a sandbox designed to protect the application against malicious users and the host from malicious applications [29]. Proof-carrying code is a mechanism that allows a host computer to determine if a program can be executed with certainty despite being provided by an untrusted source [30, 31].

31.2.4 Metering

One aspect of WSN security threat that is not often addressed as an attack is consumer access to the sensor data. As WSNs become more advanced and versatile, the notions of user access, application-specific sensor designs, and licensing of network usage will become an issue. Metering is one approach to handling these types of issues. Although many of these approaches are too computationally or memory intensive for WSNs, they provide a starting point for development of WSN techniques.

SiidTech Inc., an Oregon startup company, has proposed an approach for integrated circuit identification from random threshold mismatches in an array of addressable MOSFETs. The technique leverages on process discrepancies unavoidably formed during fabrication. This analog technique can be used in tracking semiconductor dies, authentication, and intellectual property (IP) tagging [32]. Sampling and auditing are the two main methods used for measuring the usage of media channels. Sampling conducted by Nielsen Media Research and NetRatings Inc. is based on surveys among a representative group of users [33]. Web page access metering has been addressed by a number of researchers and companies [34–36].

Licensing is the most common approach to protecting software. It provides a certain degree of control to the vendor in terms of software distribution and may prevent unauthorized duplication of software packages. The most common technique is based on the license key concept. A key is encrypted by using a string of data that contain software package ID and its usage constraints (e.g., expiration date) and the serial number of the computer where the key is installed. The invocation of the software package is done automatically when software is invoked by using one of the password schemes [37, 38]. A large number of patented licensing protocols have been proposed; for example, licenses can be used to authenticate the legal users, as well as to upgrade the products and other after-market information transmissions [39] or licensing using smart cards [40, 41].

31.3 Security Architectures

This section describes security protocols developed for two typical WSN system architectures: cell-based WSN and ad hoc WSN, with particular concentration on key establishment and distribution algorithms because they set up the necessary infrastructure for security protocols. The proposed WSN system architectures differ in many aspects, which is not surprising because WSNs operate in vastly different physical environments, supporting different applications and using different sensor nodes. The main benefit of the development of a specific architecture for each WSN application is the efficient utilization of scarce sensor node resources.

Many elements of WSN architecture, including hardware architectures of sensor nodes, routing protocols, and level of abstraction between the layers of the architecture, can be adjusted to improve performance and optimize resource consumption. One possible categorization of wireless ad hoc network systems, which includes WSN systems, is given in Law et al. [42]. Here, only the security architectures for WSN systems are discussed, while Papadimitratos and Haas [43] give an overview of security architectures for general wireless ad hoc network architectures. From the security point of view, the WSN system architectures can be broadly divided in two categories:

- Cell-based WSNs consisting of low-power low-cost sensor nodes and base stations, operating in relatively friendly environments of houses and office buildings, or in easily accessible outdoor areas
- Ad hoc WSNs consisting only of low-cost sensor nodes distributed in an ad hoc manner into remote and inhospitable environments without any wireless infrastructure

These two network architectures differ in terms of the security threats to which they are exposed and in terms of security requirements and abilities to support security architectures of various levels of complexity. The cell-based WSN allows for more sophisticated and resource-consuming protocols and algorithms because the additional computationally expensive workload can be assigned to the base stations.

31.3.1 Cell-Based WSNs

In cell-based WSN, the nodes are organized around one or more base stations that have significantly more computing and energy resources than the regular sensor nodes. These networks are most often used for user and object tracking systems in home and commercial building environments, as well as in outdoor perimeter-monitoring systems. The base stations collect information from the network and provide a link between the WSN and the outside world. Cell-based networks are often used in an environment in which it is easy to add new nodes, remove the ones that are not functioning, and even recharge the energy supplies for nodes. However, even in such an environment, the nodes can still be captured or damaged, and unauthorized nodes can be added.

The presence of base stations in a WSN offers at least two significant benefits:

- Base stations represent a trusted base that cannot be compromised. They can be used as a safe source of mobile code and configuration parameters, which enables safe bootstrapping and configuration of the network, as well as the addition of new nodes.
- Base stations offer computational resources that can be used in asymmetric security protocols in which they perform the majority of intensive computations. Such protocols allow stronger security, while not exhausting the limited resources of regular sensor nodes.

An example of a WSN organized around one or more base stations and SPINS, the security protocol suite for that network, is described by Perrig and colleagues [44]. The network consists of a trusted backbone of base stations with unlimited power supply and a large number of *motes* (low-cost, low-power sensor nodes described by Hill and colleagues [45]), distributed in the area covered by the base stations. The operation of the network is fully controlled from the base stations. A routing structure is formed as a set of routing trees; each base station is the root of one such tree. The traffic mainly consists of requests initiated at the base stations and sent down the trees to the nodes and the responses sent from the nodes back to the base stations. When the same request is sent to all nodes, the communication is most efficiently performed through broadcast messages. If a base station needs to send a unicast message to a particular node, source routing is used.

The SPINS protocol suite assumes that the base stations share a unique master key with each node in the network. The system architecture and security protocols require that the base station keep track of the route to each node and of the secret key. All other keys that the base station and a node use for communication are derived from the master key. Even though the base station is a single point of failure, it is trusted, implying no one can capture the station and recover all keys.

This security architecture efficiently uses the resources of the base stations. To keep a separate key for each node would not be possible in an architecture in which all nodes have limited resources. Also, this solution is not applicable to networks in which any two nodes are likely to communicate directly. However, because the bulk of traffic in the network is between the base station and the nodes, the inability of the nodes to communicate securely without involvement of the base station is of limited importance.

The SPINS protocols suite consists of two building blocks, sensor network encryption protocol (SNEP) and μTESLA. SNEP protects the unicast communication between the base stations and the nodes, while μTESLA provides secure broadcast communication. Each of these protocols will be discussed in more detail.

31.3.1.1 SNEP

The basic confidentiality of messages in any secure system is achieved through encryption. Encryption protects the network from adversaries who have the capability to listen to network traffic. SNEP uses

RC5 block cipher [46] for basic encryption. The original RC5 encryption algorithm is implemented with lowered functionality and generality in order to fit in the limited storage space of nodes. In addition to basic confidentiality, SNEP offers *semantic security*, which means that the encryption of the same plaintext produces a different encrypted message each time. This is achieved by keeping a shared counter on each of the two entities involved in the message exchange and incrementing the counter for each message.

Because the value of the counter is an initialization vector for the RC5 block cipher, it is guaranteed that the encrypted messages differ even if the content is the same. An additional benefit of the counters is that they ensure *freshness* of messages, i.e., a receiver can establish the partial message ordering of the messages from a particular sender. Finally, each node has its separate master key, so SNEP guarantees authentication of messages that the nodes receive from the base station.

An important property of such a solution is that it can be used in environments with relatively static forwarding structure, in which the nodes communicate with a limited number of other nodes or base stations, usually smaller than the number of neighbors. The number of the keys and counters can be estimated, and the efficiency of such a solution is known in advance. In a network with a dynamic forwarding structure in which any neighbor can be a previous or a next hop for any message, it may be prohibitively expensive to keep counters and separate keys for all possible sources and destinations. However, for a limited number of cases, two nodes that need to communicate directly can use their master keys to generate and exchange a session key through the base station.

31.3.1.2 µTESLA

The second element of the SPINS protocol suit is µTESLA. The master key shared between each node and the base station ensures confidentiality and authentication of unicast messages exchanged between the nodes and the base station. However, if the same message is sent from a base station to all nodes, it is much more efficient to broadcast the message. SNEP does not support secure broadcast because each master key is unique; allowing nodes to accept unencrypted, unauthenticated messages would allow an adversary to send arbitrary requests to nodes. Therefore, for secure broadcast communication, SPINS proposes µTESLA, the goal of which is to ensure authentication of broadcast messages sent from the base stations to the nodes.

In µTESLA, a base station generates a reverse key chain containing the keys K_0, K_1,…, K_n. The key chain length and the key K_n are determined before the key chain is generated. Other keys are determined using one-way function F, $K_i = F(K_{i+1})$. The key K_0 is not used to authenticate any of the messages, but is distributed initially as a commitment to the key chain. The distribution of the commitment K_0 in µTESLA requires that each node and the corresponding base station share a secret key unique for that node. Then, the base station sends K_0 to all nodes as a sequence of unicast messages, before any broadcast message is transmitted.

The time is divided into the intervals I_1,…, I_n, as shown in Figure 31.1, where each interval I_i corresponds to the key K_i. During the interval I_1, the base station sends broadcast messages with attached message authentication code (MAC) calculated using the key K_1. Because the key K_1 has not been disclosed yet, the messages could not have been forged by any of the nodes. The function F is a one-way function, so no one can determine K_1 from K_0. The nodes authenticate the messages received during the interval I_1 at the end of that interval, when the key K_1 is disclosed. At that time, the nodes compare K_0 with the value derived from $F(K_1)$. If the values match, then the messages authenticated with K_1 are sent from the base station, because only the base station could have known the value of K_1 before that key was disclosed. After K_1 is disclosed, the following broadcast messages are authenticated, using K_2, until K_2 is disclosed, and the process continues until the interval I_n expires.

Because the keys are disclosed in periodic intervals, the base stations and nodes must be at least loosely synchronized. If a node does not receive a message with a disclosed key and its clock is late, an adversary who received the disclosed key can forge and send messages with the MAC calculated using that key. A node with an unsynchronized clock would accept such messages for the interval equal to the delay of the node's clock.

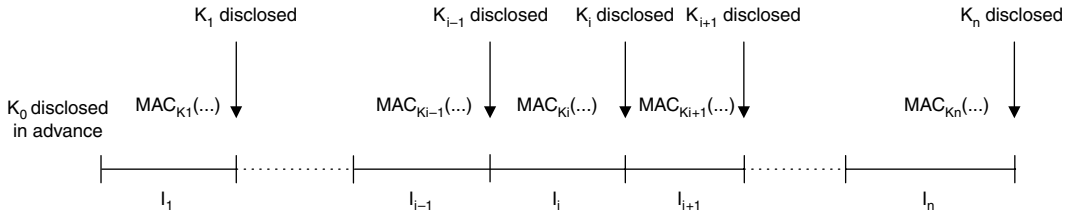

FIGURE 31.1 The key distribution timeline. The messages sent during the interval I_i have attached MAC calculated using the key K_i. The key K_i is disclosed at the end of the interval I_i.

Even if a node does not receive all the keys, it can still authenticate all messages. Once the node receives the key K_{i+1}, it can derive all previous keys $K_0,\ldots, K_i$ by successively applying the function F, and then use these keys to authenticate messages received in the corresponding intervals. However, buffering the messages for a prolonged time requires additional storage. Nodes have limited available memory, so additional mechanisms are needed to ensure that the keys are disclosed to all nodes in a timely manner.

If some nodes are deployed later and begin receiving messages during the interval I_i, they need to receive the key K_i at the end of the interval. The initialization process for those nodes is the same as for the nodes initialized when the whole network is bootstrapped; the only difference is that instead of the commitment K_0, the nodes receive the key K_{i-1} as a commitment. By doing so, new nodes save a certain amount of computation because they do not need to compute all keys from the chain $K_0,\ldots, K_{i-1}$ to compare the value of the key K_0 with the value of $F^i(K_i)$; they would need to do this if they received K_0 as a commitment. Instead, the nodes simply compare the commitment K_{i-1} with the value of $F(K_i)$.

The initial distribution of the key chain commitments requires that a base station send a separate unicast message to each node. In addition to the energy consumption of sending multiple unicast messages with the same information, the time required to initialize thousands of nodes is measured in tens of seconds, as calculated by Liu and Ning [47], who proposed that instead of costly initialization using broadcast messages, the commitment K_0 be embedded into the nodes during initialization, before deployment of the nodes. This solution brings significant savings for the majority of the nodes deployed together. For the nodes added later, there is a trade-off between computation expenses incurred when the new nodes authenticate a broadcast by comparing the embedded commitment K_0 and $F^i(K_i)$ and when the added nodes receive unicasts containing K_{i-1} and authenticating broadcast with only one calculation of the function $F(K_i)$.

The decision as to which mechanism for initialization of the added nodes is preferred could be potentially based on the number of new nodes that should be initialized. If the number is small, then the delay incurred by sending unicast messages is acceptable; however, if many nodes are added, it is more efficient to have the nodes use K_0 as a commitment. Unfortunately, the decision about the preferred commitment distribution mechanism cannot be made online because delivering the decision to the nodes would require sending authenticated unicast (which is the expense to avoid if the number of new nodes is large) or sending a nonauthenticated broadcast, which then could be a message forged by an adversary.

An implicit assumption in μTESLA is that the base stations have sufficient memory storage to hold a long key chain or that they have enough processing power to compute keys fast enough while keeping in memory only the last member of the chain. That assumption saves the nodes from buffering too many messages because the intervals can be arbitrarily short; the key chain is still long enough not to require frequent costly commitment distributions. Liu and Ning [47] propose a hierarchical organization of the keys that decreases the required memory storage at the base stations. The basic principle behind this solution is that the base stations keep only a high-level key chain in memory, while the elements of the low-level key chains are generated using the high-level keys. The keys $K_0,\ldots, K_n$ from the high-level chain are not used for message authentication. They only authenticate messages containing low-level key chain commitments $K_{<0,0>}$, $K_{<1,0>},\ldots$, $K_{<i,0>},\ldots$, $K_{<n,0>}$.

This extension of μTESLA needs to keep the property of the original scheme that even if some key disclosure messages are lost, the later key disclosures can be used to authenticate previous messages. Otherwise, the network would need to ensure that all messages are received at all nodes — an expensive proposition for WSNs. In order to enable authentication despite lost messages, the high-level key K_{i+1} is used to generate the last key for the low-level key chain for the interval I_i, $K_{<i,m>} = F_1(K_{i+1})$. Without this relation, if a message with the commitment $K_{<i,0>}$ is lost, the nodes could not authenticate the messages from the interval I_i. Even in this solution, if the message with the commitment $K_{<i,0>}$ is lost, the nodes must keep all messages from the interval I_i in a buffer until the key K_{i+1}, or some other later key, is disclosed in the interval I_{i+2}. Because the high-level intervals are intentionally kept long so that the number of keys stored in the memory of the base stations is small, the memory required to store the messages may be prohibitively large. One possible solution is to repeat messages frequently that contain key commitments.

Authentication of broadcast messages sent from the base stations to the nodes is supported by μTESLA. It may be possible to use the protocol in cell-based networks in which the nodes send broadcast messages too. Two possible solutions for this problem are: (1) a node sends a unicast to the base station using the key that the node and base station share, and then the base station broadcasts the message using the original μTESLA broadcast authentication mechanism; or (2) a node broadcasts the message and the base station handles the distribution of a key for that broadcast.

31.3.2 Ad Hoc Sensor Networks

Certain military, law enforcement, and disaster recovery WSNs are deployed in remote and inhospitable environments without any wireless infrastructure. Nodes must self-organize and bootstrap a network without any support from base stations. Such networks distributed in an ad hoc fashion must be capable of accepting requests from various points within the network because a user walking through the area may not have the capability to connect to the designated gateway. Any node in such an architecture can be a source of or a destination for messages.

Even more than in other networks, the nodes in such systems are exposed to a danger of capture or destruction. The most dangerous physical threat regarding security is physical possession of a sensor node by an adversary. Sensor nodes may contain keys that allow the adversary to decrypt the messages and even to inject false messages into the network. In circumstances in which long-term security of all nodes in a network cannot be guaranteed, the best solution is to extend the lifetime of the network as much as possible. There are two aspects of extending the lifetime of a network:

- The time period from when the network is deployed to the moment a node is compromised should be as long as possible. An adversary can determine the positions of nodes using various technologies. The easiest way is to listen to the messages exchanged between the nodes because they usually contain the locations of nodes that detected an event. In Slijepcevic et al. [48], messages are encrypted with a separate encryption algorithm for locations of nodes; this is stronger than the encryption for the rest of the message content so that the adversary has less encrypted text for cryptanalysis. If the information about the locations of nodes is adequately protected, the adversary is left using trilateration, which requires more equipment and effort than simply extracting the locations of nodes from the messages.
- Once some of the nodes are detected, the keys that these nodes contain can be extracted and used to decrypt previously exchanged messages as well as future ones. Key distribution mechanisms in WSN and secure protocols must be designed so that the security exposure is minimized when any of the cryptographic keys is compromised.

In a system architecture in which all nodes are potential senders or receivers, symmetric cryptography suits the low-power nodes better than public cryptography. Because symmetric cryptography assumes that the keys are shared, the design space between two extreme solutions remains: (1) all nodes share only one key embedded in them before the deployment; and (2) each pair of nodes shares a unique key.

The first solution is simple, does not require too much memory space, and has the broadcast primitive available. However, when one node is compromised, the adversary can decrypt all messages from the network. The second solution has a perfect security property: if a node is compromised, the recovered keys are useless because no other nodes use those keys. However, the memory space for all keys for networks of thousands of nodes is not available on most sensor node platforms. Even if only a handful out of thousands of keys is actually used when a network is deployed, the nodes must store them all because their exact physical locations are not known before the deployment, and they cannot know which nodes will be located close to each other.

Additionally, sending broadcast messages is not possible, so each broadcast message must be replaced with multiple unicast messages, and the energy consumption is multiplied accordingly. The key distribution algorithms proposed for WSNs try to find a trade-off among the various requirements. The important factors for the key distribution algorithms are:

- Impact of compromise of one or more nodes on security of the traffic in the network
- Ability of algorithm to include additionally deployed nodes into the security infrastructure
- No single point of failure
- Spatial and temporal variation in keys to reduce encrypted material for cryptanalysis
- Support for broadcast

31.3.2.1 Key Distribution Schemes

Most key distribution mechanisms shy away from key distribution after the nodes are deployed. Such schemes exist for various wired and wireless networks and they mainly include key distribution servers. They consider self-organized wireless network with no security infrastructure. Therefore, no central authority, no centralized trusted party, and no other centralized security service provider exist. The standard solutions for authenticated broadcast are not applicable. The solution used in wireless networks with more capable nodes [49, 50] employs public key cryptography to ensure authenticity of messages. However, in many WSN projects [44, 48, 51], the public key algorithms are considered too expensive in terms of memory and processing requirements to be used in WSNs, except as a one-time protocol for exchange of private keys. A thorough discussion about the energy requirements of the public key encryption algorithms and their performance on various processors is given by Yuan and Qu [52].

A possible solution for key distribution is to assign some nodes to be key distribution servers, delivering symmetric keys to nodes that need to communicate. The use of online key distribution servers in WSN has a disadvantage; because all nodes are physically exposed, key distribution servers would become single points of failure because of failures and especially because they would allow an adversary to get hold of all keys used in networks. Therefore, the key distribution algorithms presented here are based on key assignment before deployment. The addition of new nodes and the loss of previous nodes does not require an immediate key distribution process, as is the case in Internet multicast algorithms in which new nodes must not be able to read previously exchanged messages, and the old nodes must not be able to read future messages. In WSNs, new nodes are trusted and old nodes are most likely out of energy.

Eschenauer and Gligor [51] propose a probabilistic key distribution in which a node shares a key with a certain percentage of other nodes. Before the deployment, an initial pool of P keys is generated. For each node, k keys are selected from the initial pool for a key ring. After the deployment, the nodes announce and compare their key rings, looking for at least one key that belongs to both key rings. If such a key is found, those two nodes can communicate directly. When all such pairs are found, they represent the connectivity graph for the network. Now, even the pairs of neighboring nodes that do not have a direct connection can use an established secure path to generate a key, or pick a key from a set of unused keys from the key rings and exchange that key. Eventually, each pair of neighboring nodes will share the key, under the condition that the network graph was connected initially. If that was not the case, a certain number of nodes are permanently excluded from the network.

The main advantage of this scheme is its resiliency in the case of compromised keys. If a node is captured, all of its k keys are available to the adversary. The probability that a particular key is used for

encryption of a link is the same for all keys, so a probability that the adversary can decrypt traffic on a particular link is k/P. As will be explained later, k is significantly smaller than P; thus, that probability is low. The scheme also achieves significant memory savings compared to a scheme in which each pair of nodes shares a key. Furthermore, when new nodes are added to the network, they announce their key rings in the same way as the nodes deployed during the initialization of the network.

Because different keys are used throughout the network, the amount of encrypted material for cryptanalysis is smaller in such a key distribution scheme than in the scheme with a shared key for all nodes. If the lifetime of the network is so long that the keys should be replaced, the nodes can revoke the keys with the expired lifetime. After some keys are revoked, the process of establishing secure connections must be run again, with fewer keys. The removal of keys decreases the probability that the network will be fully connected, so the key revocation has limited usage. Finally, the scheme does not support broadcast. However, after a node establishes secure paths to all its neighbors, it can distribute one of its keys as a broadcast key in the case of increased broadcast traffic. Obviously, the applications running on top of the WSN running this key distribution scheme need a certain amount of control over the deployment of certain mechanisms; such mechanisms are not always needed, but their deployment consumes energy.

The parameters of the scheme are determined as a trade-off between security in the case of compromised nodes and the probability that the network is connected. The parameter k, the size of a key ring, is determined by the size of the memory reserved for the keys. The size of the network and the average number of nodes within a communication range are determined by the network application and topology. The only parameter that can be changed over a large range of values is the size of the initial key pool, P. If P decreases, the probability that two key chains selected from the keys from P have one or more common keys increases. However, the value of the expression k/P, which represents the probability that an adversary can compromise a communication link when a node is compromised, also increases.

A result from graph theory, presented by Spencer [53], determines the probability, p, that an edge is between two vertices in a graph, for which the probability that the graph is connected rises from a small probability to "certainly true." Then, P is determined from the condition that the probability of two key chains with one or more common keys is equal to p. For a network of 10,000 nodes, with $d = 40$ neighbors per node on average and the size of a key ring $k = 15$, if the size of the initial pool is $P = 100{,}000$, the network is fully connected with the probability .99999.

Chan and colleagues [54] offer two improvements to the described scheme. The first change is that two nodes can establish a secure link only if they share q keys, instead of one as in the original scheme. The advantage of this approach is that, for a small number of captured nodes, the probability that any link in the network can be compromised is lower than if the nodes establish a link with only one shared key. However, with the increased number of captured nodes, the relation between these two probabilities changes, so with a sufficiently high number of captured nodes, an adversary has better chances of compromising the secure links than in the original scheme. This trade-off improves the protection of the network against small-scale attacks, which are easier to execute and therefore more likely, and decreases the protection against larger attacks, which are more expensive to perform.

The second improvement from Chan and colleagues [54] allows pairs of nodes that have a secure link between each other to establish new keys. That way, more keys are used, while the amount of the memory required to hold the keys is kept low, at the order of magnitude of the number of neighbors. The price paid for this improvement is increased traffic for key exchanges. It is also important to mention that the two schemes proposed here should not be used at the same time. The first scheme requires that the number of keys in the initial pool be kept low to ensure the connectivity of the network. At the same time, the second scheme tries to use different keys; however, during the exchange of the keys the probability of capture of these keys is increased.

31.4 Privacy Protection

The previous sections have examined the security architectures for two broadly defined types of WSN. The main goal of the presented architectures is to establish secure communication channels within a network in order to protect transmitted information from unauthorized access. In many WSNs, especially in military and law enforcement systems, sensor nodes and communication between them are the most exposed part of the network. In such networks, reliable and secure communication is the most important and best guarantee of uninterrupted functionality.

For another class of WSN systems — those intended for use in commercial settings — the privacy protection of individuals observed by a WSN whose living and working spaces are populated by sensor nodes is as important as the protection of applications' functionality. It is still necessary to ensure secure communication channels in commercial WSNs in order to prevent unauthorized access to the personalized information about the users of the system. However, even if the communication security architecture ensures that the personal information is well protected during transfer through the network, once such information is collected at a data collection point, the information is protected as much as the data collection hosting system is. Commercial systems tend to have lower standards for security protection of acquired information than military and law enforcement systems do. The news frequently reports about systems in which system security at data collection points was compromised and social security numbers, credit card numbers, and many other highly sensitive and personal data ended up publicly available to anyone on the Internet. These cases illustrate the need for additional mechanisms that will ensure a certain level of privacy protection without interfering with the functionality of commercial WSN systems.

31.4.1 Principle of Minimal Generalization

Sensor nodes' sensing capabilities, size, and low cost allow a large number of sensor nodes to be deployed in and a large amount of information to be acquired from physical surroundings. Except in rare cases, the larger the amount and the higher the precision of the sensing data available to a WSN, the better the performance of applications is. Although the applications may perform better if more data are available, privacy protection, by definition, strives for the minimum amount of data to be acquired about a single individual. Although the performance of applications and the need for privacy protection may seem to be two opposing goals, many applications can function effectively with their information precision at a lower level than the level of precision that WSNs are capable of delivering. That interval between the required and potential accuracy can be effectively used for privacy protection.

Samarati and Sweeney [55] have proposed a mechanism for generalization of data in databases in order to prevent matching individuals and their medical records. The medical records with the names removed, but with ZIP codes, dates of birth, and other information, are easily matched with the identities acquired from voter lists, city directories, and other publicly available sources. To prevent reidentification, nonessential information is removed, i.e., the year of the birth is kept, while the exact date is removed. The goal of the process is to depersonalize medical records so that multiple identities are equally likely to correspond to a particular medical record. This approach is called the principle of *minimal generalization*. The same principle can be applied in the context of the privacy protection in WSNs. Naturally, this may affect applications that operate on generalized data, so the principle can be applied only if the application can retain the required performance level. An informal description of the principle of minimal generalization is: accurate private information about the users of a system should be generalized so that the acquired data can be matched to no less than k identities, where k is the required level of anonymity.

Under the assumption that a data collection point and a WSN are under separate control, this principle is beneficial for both entities. The WSN offers higher privacy protection for its users, while the data collection system does not need to expend resources for privacy protection purposes and does not need to risk liability for possible breaches of privacy protection.

31.4.2 Privacy of Location Information

This principle is demonstrated on several WSN applications that rely on location information about users. Protection of the location information is highlighted for three main reasons:

- The most frequent tasks for WSN systems are concerned with detection of location of an event. Even if the goal of an application is to perform a more complex task, the location information is present as a part of the individual observations generated by sensor nodes.
- The privacy protection of location information for users observed by a WSN is a prime example of the importance of data protection because, with access to the location data for a user, an adversary can infer additional private information — for example, medical conditions, shopping habits, and patterns of social interactions between monitored users.
- Protection of location information allows the principle of minimal generalization to be demonstrated in an easily understandable case study. Generally, location discovery systems are often capable of locating users within meters indoors and within tens of meters outdoors. For many applications, that level of precision is more than required for basic functionality, so it is acceptable to reduce the precision of the information in order to achieve a required level of privacy protection.

The general system architecture for which privacy protection solutions are described is shown in Figure 31.2. The crucial part of the privacy protection framework for WSNs is the location server. The server is a part of the trusted zone, which in this context means that the server adheres to the same security policies and is controlled by the same entity as the accompanying network of sensor nodes. In fact, the location server is likely to be implemented as a service running on a gateway between the sensor network and the outside world. The assumption that sensor nodes are trusted and that they do not forward any information to an unauthorized party extends here to a location server. The responsibility of the location server is to transform the locations of users observed by the network into a representation that keeps the level of location privacy protection above a certain threshold. The main difference between various privacy protection algorithms is in the types of transformations performed by a location server. The transformed location information is then forwarded to any of the servers offering location-based services (LBS). The services are offered by various entities that do not share security trust with the WSN.

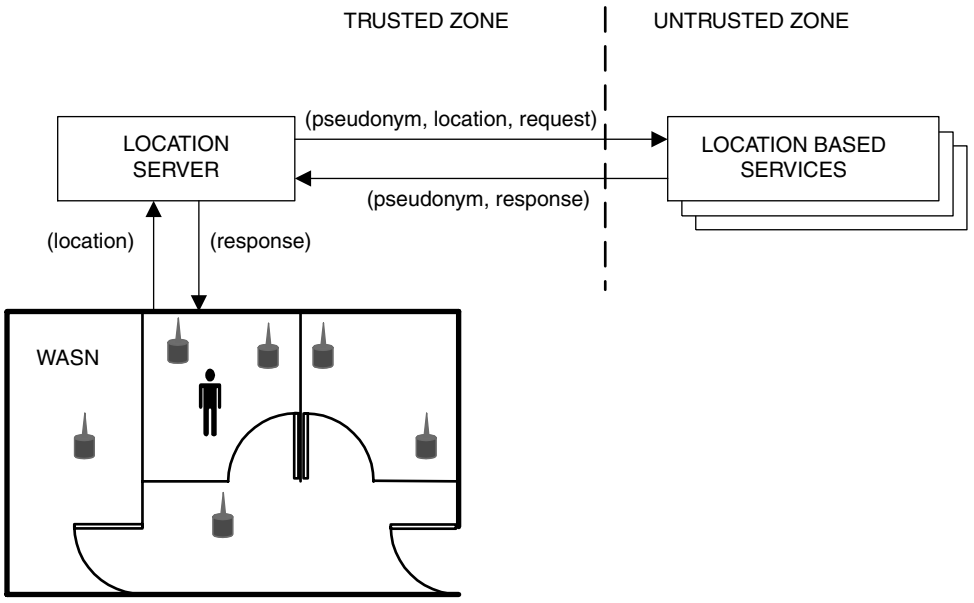

FIGURE 31.2 The architecture of the system connecting a wireless sensor network and a location-based service. The location server transforms location information, so the identities of users are hidden.

Many WSN and wireless network applications may run on top of a system architecture similar to the one shown in Figure 31.2. Two applications, which offer road maps and road condition information based on the location of a car, are proposed in several projects [56] and are commercially available [57–59]. In Beresford and Stajano [60], various proposed applications are implemented on the top of an indoor location system in an office building environment. Gruteser and coworkers examine the privacy concerns of applications that track the use of different building areas [61].

In all these applications, users send information about their locations to an LBS to update their locations or to request services offered in their vicinity. Without transformation performed in a location server, each user request or update would be accompanied with as precise location information as the location discovery technology used allows. In automotive applications, precision is defined by the precision of a GPS receiver, while in an indoor environment location precision depends on the density of the sensor network, usually precise enough to locate a room where a person is correctly located. If the information from these location discovery services is compromised, the location precision allows for easy recovery of users' movements by LBS.

The first step to protect users' privacy is to disconnect the location information from the explicit user identification. The location server performs this task by assigning an alternative identification or a *pseudonym* to each user. Some kind of identification is necessary because a response to each request among a possibly large number of requests handled by a location server must be forwarded to the original user. For many LBSs, it is not necessary for a service to be aware of a user's real identity. A road map can be generated for a user based only on the user's current location. However, two problems occur with privacy protection through anonymity of identifications. The first is the possibility that if a user uses the same pseudonym when connecting to various LBSs, the combined data from all LBSs can give a full overview of that user's activities. That problem can be solved simply by using different pseudonyms for various LBSs, similar to a solution for the same problem outside the context of WSNs, as proposed by Chaum [62].

The second problem is that a user can be easily identified despite different pseudonyms, if requests for LBSs are coming from specific locations that can be directly connected to the user. In the case of a request for a road map that an LBS issued from a location that can be identified as a private garage, the anonymous identification can be attached to the owner of the garage, and then all the movements of that ID can be personalized. In the same way, in an office building environment, an ID that spends most of the time in a particular office can be connected to the regular occupant of that office.

The solution for this problem is in the combination of the principle of minimal generalization and temporary anonymous identifications. Before details of the mechanism are described, it is necessary to include a more formal definition of privacy in order to be able to compare the benefits of different approaches to this problem. The measure of privacy in this context is the notion of *k-anonymity*. The meaning of the term *anonymity* in privacy protection research is formally defined by Pfitzmann and Koehntopp [63]; they define it as a quality of not being identifiable within an anonymity set containing a set of subjects. Then *k*-anonymity, as defined by Samarati and Sweeney [55], is anonymity within a set with the cardinality of *k*. This term is used to define an acceptable level of privacy protection in which a person cannot be distinguished from $k - 1$ other individuals. For the application using location information, *k*-anonymity means that the attached location information for that user comprises location information for $k - 1$ other users. If a stretch of a freeway of the length d contains k cars, a user whose location is defined with the resolution d is *k*-anonymous.

Gruteser and Grunwald [56] achieve *k*-anonymity by transforming precise, high-resolution GPS–originated location information available to a location server to low-resolution location information sent to the LBS. Additionally, the identity of a user is hidden in order to avoid continuous tracking of his location. If a user's location is defined by the intervals $[x_1,x_2]$ and $[y_1,y_2]$ in two-dimensional space and the time interval $[t_1,t_2]$ in time dimension, *k*-anonymity is achieved by extending and contracting the intervals until $k - 1$ or more objects share the resulting parallelepipe in the three-dimensional space-time coordinate system. Depending on the nature of the application, Gruteser and Grunwald have proposed two different algorithms [56] that transform resolution of location information:

- If the application requires a timely response, the spatial resolution is brought to a level at which k-anonymity is achieved. The implementation of this algorithm starts from the entire area covered by the LBS. The area is then divided into subareas, until the subarea containing the specified user also contains less than $k - 1$ other users. Examples of such applications are road map and road condition LBSs in which the delay must be on the level of minutes; otherwise, the information returned from the LBS cannot be effectively used.
- If a delay is acceptable, the application can set a threshold on the spatial resolution, requiring that the area confined with the intervals $[x_1,x_2]$ and $[y_1,y_2]$ is never above the given threshold. The property of k-anonymity for a user is then achieved by extending the temporal interval $[t_1,t_2]$ by simply waiting until k or more users pass through the space limited by the spatial intervals. Now, the LBS side of the application deals with a more precise location information, which is beneficial for the quality of service that LBSs offer.

In Gruteser and colleagues [61], the underlying application counts the number of people in various parts of a building to estimate the utilization of the rooms in the building. The nodes in the network are organized in a hierarchical structure, with sensor nodes at the lowest level detecting individuals in their vicinity, usually only in one part of a room. The nodes at the next level count the number of individuals in each room, using the sensing data from the nodes from the lowest level. The hierarchy structure assumes nodes at the floor level as the next level and then, finally, a location server that gathers the data from the floor level. Without privacy protection, the information about occupancy of the rooms would be simply transferred up to the location server. The application requirements are such that it can perform its task without identifying individuals occupying rooms. However, similar to the applications mentioned previously, if an adversary can access the data gathered at a data collection point, he can reidentify individuals from the rooms that each of them occupies most frequently. Now, identified individuals can be tracked by observing counterchanges in various parts of the building.

The solution proposed in this work leverages the hierarchical network architecture of the WSN [61]. A node determines a count of individuals in its area, by sensing, if at the lowest level of the system hierarchy, or by aggregating the counts received from the nodes one level below, if at one of the higher levels. The node then compares the count with a threshold value k. If the value is below k, that value is propagated to a higher level with decreased resolution of the location information. If the value is above k, the value sent to the upper level is the nearest multiple of k. In that case, the location information is accurate. Using this algorithm, even a location server does not need to belong to a trusted zone because the obfuscation of the location information is already performed in the network.

The work in Beresford and Stajano [60], in which anonymous users are tracked through a building, notes the same problem with permanent anonymous IDs that can be connected to a particular user based on location information from a private area. However, the applications from that work cannot allow for a location precision coarser than the level of a room, so the increased resolution is not an acceptable solution. On the other hand, at the room level, the location information can easily be used to find out which identifiers spend the most time in a particular room.

These authors propose a solution in which the concept of k-anonymity is used in special areas called *mix zones* where users change their temporary anonymous IDs [60]. The manipulation of identifiers is a responsibility of a location server. However, the authors also note an important weakness of the solution with *mix zones*. They demonstrate that the initial assumption that, if two individuals cannot be tracked when they enter a *mix zone* from opposite directions and then reappear from them with different pseudonyms does not hold well if an adversary uses a statistical analysis. The experimental results show that the probability that each individual will return to the same direction from which he came is 1%, so if an adversary assumes that a user who entered from one direction and the user who left the mix zone at the other end are one person, regardless of IDs, he will be correct in 99% of cases.

There are certainly applications that do not conform completely to the system architecture from Figure 31.2. The possible differences are the expansion of the trusted zone to include an LBS, in which case a location server is not needed. The example of this type of application is provided in Priyantha et al. [64]:

indoor user location detection system. The beacons embedded in the building transmit the information about their locations. A device carried by a user detects the signals from beacons, and then determines the location of a user from multilateration of distances acquired from the signal strengths of beacons' signals. In this case, the information about the location is kept on the user's personal device, so privacy is not a concern. However, this simple case has a downside because the user must perform additional work in order to match his location to the location of an interesting object or service. Such a solution is possible only for applications in which results are stored at a device controlled by a user.

Finally, in some applications, it is necessary to maintain relationships between an individual and his profile at a data collection point on an LBS. An example of such an application is Networkcar service [65]. A network of sensors in a car checks the state of the engine and other functioning units in a car. Each car has a built-in gateway that connects the car with a mobile telephony network. At the same time, the gateway uses a GPS client to determine the position of the car. All this information is stored on a Web server. A user of the service (an owner or an authorized car mechanic) can log on to the service through the Web and examine the current location of the car, the conditions of its engine, and other information. The owner of the car receives a message if it has been stolen and taken out of a certain area. This type of application cannot use the proposed techniques in which the precision of location information is reduced because the continuous connection between user and location information sent to the database must be maintained.

31.5 Conclusion

For many military and civilian applications of wireless sensor networks, security and privacy protection protocols and algorithms are an indispensable part of the system architecture. Because of their unique properties, most notably limited resources and physical exposure of sensor nodes, sensor networks require a new type of security protocols. These protocols are tailored to the underlying system architecture, patterns of network traffic, and specific security requirements so that security-related resource consumption is minimized. Physical exposure of nodes, as well as the threat that their cryptographic secrets are potentially available to an adversary, demands that security protocols in sensor networks protect the integrity of the network even if cryptographic secrets are compromised.

Privacy protection is especially important for certain commercial applications of sensor networks. Users who are monitored by sensor networks expect their private information not to be publicly available. However, sensor networks need services from other entities that may not have satisfactory privacy protection mechanisms. In cases in which applications require less precise data than are available, a certain level of privacy protection can be achieved by decreasing precision of the data; therefore, the data cannot be easily matched to any particular individual.

References

1. Industrial automation, Ember Corp. (2003). Retrieved August 20, 2003, from http://www.ember.com/products/solutions/industrialauto.html.
2. Dynamic sensor networks. Retrieved September 1, 2003, from http://dsn.east.isi.edu/.
3. Anderson, R. and Kuhn, M., Tamper resistance — a cautionary note, in *Proc. 2nd USENIX Workshop Electron. Commerce*, USENIX, Berkeley, CA, 1996, 1.
4. Stajano, F. and Anderson, R., The resurrecting duckling: security issues for ad-hoc wireless networks, in *Proc. 7th Int. Workshop Security Protocols*, Christianson, B., Crispo, B., and Roe, M., Eds., Springer-Verlag, Heidelberg, 1999, 172.
5. Meguerdichian, S. et al., Localized algorithms in wireless ad-hoc networks: location discovery and sensor exposure, in *Proc. 2nd ACM Symp. Mobile Ad Hoc Networking and Computing* (MobiHOC), ACM Press, New York, 2001, 106.

6. Krishnamachari, B., Estrin, D., and Wicker, S., The impact of data aggregation in wireless sensor networks, in *Proc. Int. Workshop Distributed Event-Based Systems*, IEEE Computer Society, Los Alamitos, CA, 575.
7. Cheswick, B., An evening with Berferd in which a cracker is lured, endured, and studied, in *Proc. of Winter USENIX Conf.*, USENIX, Berkeley, CA, 1992, 163.
8. Stoll, C., *The Cuckoo's Egg*, Doubleday, New York, 1989.
9. Bellovin, S., Leech, M., and Taylor, T., ICMP traceback messages, Internet draft (work-in-progress), IETF, 2003.
10. Savage, S. et al., Practical network support for IP traceback, in *Proc. ACM SIGCOMM Conf. Applications, Technol., Architectures, Protocols Computer Commun.*, ACM Press, New York, 2000, 295.
11. Snoeren, A. et al., Single packet IP traceback, *IEEE/ACM Trans. Networking*, 10(6), 1, 2002.
12. Song, D. and Perrig, A., Advanced and authenticated marking schemes for IP traceback, in *Proc. 20th Joint Conf. IEEE Computer Commun. Soc.* (INFOCOM), IEEE, 2001, 878.
13. Boulis, A. and Srivastava, M.B., A framework for efficient and programmable sensor networks, in *Proc. 5th IEEE Conf. Open Architectures Network Program.* (OPENARCH 2002), New York, June 2002.
14. Cohen, F., Computer viruses, theory and experiments, *Computers Security*, 6(1), 22, 1987.
15. Kephart, J.O. and White, S.R., Measuring and modeling computer virus prevalence, in *Proc. IEEE Computer Soc. Symp. Res. Security Privacy*, IEEE Computer Society, Los Alamitos, CA, 1993, 2.
16. Spafford, E.H., Computer viruses — a form of artificial life? in *Artificial Life II*, Langton, C.G. et al., Eds., Addison–Wesley, Redwood City, CA, 1992, 727.
17. Denning, D.E., *Cryptography and Data Security*, Addison–Wesley, Redwood City, CA, 1982.
18. Neumann, P., *Computer-Related Risks*, Addison–Wesley, Redwood City, CA, 1995.
19. Cowan, C. et al., Buffer overflows: attacks and defenses for the vulnerability of the decade, *DARPA Inf. Survivability Conf. Exposition*, IEEE Computer Society, Los Alamitos, CA, 1999, 1119.
20. Wagner, M.G., Robust watermarking of polygonal meshes, in *Geometric Modeling Process.*, IEEE Computer Society, Los Alamitos, CA, 2000, 201.
21. Lampson, B.W., A note on the confinement problem, *Commun. ACM*, 16(10), 613, 1973.
22. Gold, B.D. et al., A security retrofit of VM/370, in *AFIPS Conf. Proc.*, AFIPS Press, 1979, 335.
23. Golomb, S.W., The limiting behavior of the Z-channel, *Trans. Inf. Theory*, 26(3), 372, 1980.
24. Hu, W., Reducing timing channels with fuzzy time, in *Proc. IEEE Computer Soc. Symp. Res. Security Privacy*, IEEE Computer Society, Los Alamitos, CA, 1991, 8.
25. Simmons, G.J., The history of subliminal channels, *IEEE J. Selected Areas Commun.*, 16(4), 452, 1998.
26. Rubin, A.D. and Geer, D.E., Mobile code security, *IEEE J. Internet Computing*, 2, 30, 1998.
27. Gong, L. et al., Going beyond the sandbox: an overview of the new security architecture in the Java development kit 1.2, in *Proc. USENIX Symp. Internet Technol. Syst.*, USENIX, 1997, 103.
28. Brigner, P., Creating signed, persistent Java applets, *Dr. Dobb's J.*, 24, 82, 1999.
29. Bowen, R. et al., Building survivable systems: an integrated approach based on intrusion detection and confinement, in *Proc. DARPA Inf. Security Symp.*, 2000, 1084.
30. Colby, C. et al., A certifying compiler for Java, *SIGPLAN Notices*, 35, 95, 2000.
31. Necula, G.C., Proof-carrying code, in *Proc. 24th ACM SIGPLAN-SIGACT Symp. Principles Programming Languages*, ACM Press, 1997, 106.
32. Lofstrom, K., Daasch, W.R., and Taylor, D., IC identification circuits using device mismatch, in *Proc. Int. Solid-State Circuits Conf.*, IEEE, 2000, 372.
33. Pitkow, J., In search of reliable usage data on the WWW, *Comp. Networks ISDN Syst.*, 29, 1343, 1997.
34. Franklin, M.K. and Malkhi, D., Auditable metering with lightweight security, *J. Comp. Security*, 6(4), 236, 1998.
35. Naor, M. and Pinkas, B., Secure accounting and auditing on the Web, *Comp. Networks ISDN Syst.*, 30, 541, 1998.

36. Rivest, R.L., Electronic lottery tickets as micropayments, in *Proc. Int. Conf. Financial Cryptography*, Hirschfeld, R., Ed., Springer–Verlag, Heidelberg, 1997, 307.
37. Findley, R. and Dixon, R., Dual smart card access control electronic data storage and retrieval system and methods, U.S. patent 5629508, 1997.
38. Menezes, A.J., Oorschot, P.C., and Vanstone, S.A., *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1997.
39. Ross, C.D. et al., Method and apparatus for electronic licensing, U.S. patent 5553143, 1996.
40. Aura, T. and Gollmann, D., Software license management with smart cards, in *Proc. USENIX Workshop Smartcard Technol.*, USENIX Association, Berkeley, CA, 1999, 75.
41. Thomas, D.C., Method and apparatus for providing security or computer software, U.S. patent 4446519, 1984.
42. Law, Y.W., Etalle, S., and Hartel, P.H., Assessing security-critical energy-efficient sensor networks, in *Proc. 18th IFIP TC11 Int. Conf. Inf. Security Privacy Age Uncertainty* (SEC), Gritzalis, D. et al., Eds., Kluwer Academic Publishers, Boston, MA, 2003, 459.
43. Papadimitratos, P. and Haas, Z.J., Securing mobile ad hoc networks, in *Handbook of Ad Hoc Wireless Networks*, Ilyas, M., Ed., CRC Press, Boca Ration, FL, 2002.
44. Perrig, A. et al., SPINS: security protocols for sensor networks, in *Proc. 7th Int. Conf. Mobile Computing Networking* (MOBICOM), ACM Press, New York, 2001, 189.
45. Hill, J. et al., System architecture directions for network sensors, in *Proc. 9th Int. Conf. Architectural Support Programming Languages Operating Syst.* (ASPLOS), ACM Press, New York, 2000, 93.
46. Rivest, R.L., The RC5 encryption algorithm, in *Proc. 2nd Workshop Fast Software Encryption*, Preneel, B., Ed., Springer–Verlag, Heidelberg, 1995, 86.
47. Liu, D. and Ning, P., Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks, in *Proc. 10th Symp. Network Distributed Syst. Security*, Internet Society, Reston, VA, 2003, 263.
48. Slijepcevic, S. et al., On communication security in wireless ad-hoc sensor networks, in *Proc. 11th IEEE Int. Workshops Enabling Technol.: Infrastructure Collaborative Enterprises* (WETICE), IEEE Computer Society, Los Alamitos, CA, 2002, 139.
49. Hubaux, J.P., Buttyan, L., and Capkun, S., The quest for security in mobile ad hoc networks, in *Proc. 2nd ACM Symp. Mobile Ad Hoc Networking Computing* (MobiHOC), ACM Press, New York, 2001, 146.
50. Carman, D.W., Matt, B.J., and Cirincione, G.H., Energy-efficient and low-latency key management for sensor networks, in *Network Assoc. Labs Advanced Security Res. J.*, 5(1), 2003, 31.
51. Eschenauer, L. and Gligor, V.D., A key management scheme for distributed sensor networks, in *Proc. 9th ACM Conf. Computer Commun. Security*, ACM, New York, 2002, 41.
52. Yuan, L. and Qu, G., Design space exploration for energy-efficient secure sensor networks, in *Proc. IEEE Int. Conf. Application Specific Syst., Architectures, Processors* (ASAP), IEEE Computer Society, Los Alamitos, CA, 2002, 88.
53. Spencer, J.H., *The Strange Logic of Random Graphs*, 1st ed., Springer-Verlag, Heidelberg, 2000.
54. Chan, H., Perrig, A., and Song, D., Random key predistribution schemes for sensor networks, in *Proc. 2003 IEEE Symp. Security Privacy*, IEEE Computer Society, Los Alamitos, CA, 2003, 197.
55. Samarati, P. and Sweeney, L. Protecting privacy when disclosing information: *k*-anonymity and its enforcement through generalization and suppression, technical report, SRI International, 1998.
56. Gruteser, M. and Grunwald, D., Anonymous usage of location-based services through spatial and temporal cloaking, in *Proc. ACM/USENIX Int. Conf. Mobile Syst., Applications, Services* (MOBISYS), USENIX, Berkeley, CA, 2003.
57. Navigation Technologies, Navtech. Retrieved September 1, 2003, from http://www.navtech.com.
58. Avis Assist, Avis, Inc. Retrieved September 1, 2003, from http://www.avis.com/AvisWeb/JSP/US/en/deals/us_assist.jsp.
59. Autodesk Location Services, Autodesk (2003). Retrieved August 9, 2003, from http://locationservices.autodesk.com/.

60. Beresford, A. and Stajano, F., Location privacy in pervasive computing, *IEEE Pervasive Computing*, 2 (1), 46, 2003.
61. Gruteser, M. et al., Privacy-aware location sensor networks, in *Proc. 9th USENIX Workshop Hot Topics Operating Syst.* (HotOS), USENIX, Berkeley, CA, 2003.
62. Chaum, D., Security without identification: transaction systems to make Big Brother obsolete, *Commun. ACM*, 28 (10), 1030, 1985.
63. Pfitzmann, A. and Koehntopp, M., Anonymity, unobservability, and pseudonymity — a proposal for terminology, in *Proc. Workshop Design Issues Anonymity Unobservability*, Federrath, H., Ed., Springer-Verlag, Heidelberg, 2001, 1.
64. Priyantha, N.B., Chakraborty, A., and Balakrishnan, H., The Cricket location support system, in *Proc. 6th Int. Conf. Mobile Computing Networking* (MOBICOM), ACM Press, New York, 2000, 32.
65. Networkcar technology, Networkcar, Inc. Retrieved August 21, 2003, from http://www.network-car.com.