Chapter 1

KEY DISTRIBUTION TECHNIQUES FOR SENSOR NETWORKS

Haowen Chan, Adrian Perrig, and Dawn Song
Carnegie Mellon University
{haowenchan, perrig, dawnsong}@cmu.edu

Abstract. *This chapter reviews several key distribution and key establishment techniques for sensor networks. We briefly describe several well known key establishment schemes, and provide a more detailed discussion of our work on random key distribution in particular.*

Keywords:. Sensor network, key distribution, random key predistribution, key establishment, authentication.

1. Introduction

The general *key distribution* problem refers to the task of distributing secret keys between communicating parties to provide security properties such as secrecy and authentication.

In sensor networks, key distribution is usually combined with initial communication establishment to bootstrap a secure communication infrastructure from a collection of deployed sensor nodes. In the setting we study in this chapter, nodes have been pre-initialized with some secret information before deployment, but only after network setup, we know the location of nodes. The node location often determines which nodes need to establish a cryptographic keys with which other nodes, so we cannot set up these keys before deployment. In this chapter, we refer to the combined problem of key distribution and secure communications establishment as the *security bootstrapping problem*, or simply the *bootstrapping problem*. A bootstrapping protocol must not only enable a newly deployed sensor network to initiate a secure infrastructure, but it must also allow nodes deployed at a later time to join the network securely. This is a challenging problem due to the many limitations of sensor network hardware and software.

In this chapter, we discuss and evaluate several well-known methods of key distribution. Besides these, we present an in-depth study of *random key pre-*

distribution, a method that has recently attracted significant research attention, and we have also worked on.

2. Sensor network limitations

The following characteristics of sensor networks complicate the design of secure protocols for sensor networks, and make the bootstrapping problem challenging.

- *Vulnerability of nodes to physical capture.* Sensor nodes may be deployed in public or hostile locations (such as public buildings or battle areas). The requirement for low-cost sensor nodes is in conflict with making them robust to tampering. This exposes sensor nodes to physical attacks by an adversary. In the worst case, an adversary may be able to undetectably take control of a sensor node and compromise the cryptographic keys.
- *Lack of a-priori knowledge of post-deployment configuration.* If a sensor network is deployed via random scattering (e.g., from an airplane), the sensor network protocols cannot know beforehand which nodes will be within communication range of each other after deployment. Even if the nodes are deployed by hand, the large number of nodes involved makes it costly to pre-determine the location of every individual node. Hence, a security protocol should not assume prior knowledge of which nodes will be neighbors in a network.
- *Limited bandwidth and transmission power.* Typical sensor network platforms have limited bandwidth. For example, the UC Berkeley Mica platform's transmitter has a bandwidth of 10 Kbps. Transmission reliability can be low [30], which makes the communication of large blocks of data expensive.

3. Requirements for Bootstrapping Security in Sensor Networks

A bootstrapping scheme for sensor networks needs to satisfy the following requirements:

- Deployed nodes must be able to establish secure node-to-node communication.
- Additional legitimate nodes deployed at a later time can form secure connections with already-deployed nodes.
- Unauthorized nodes should not be able to gain entry into the network, either through packet injection or masquerading as a legitimate node.

- The scheme must work without prior knowledge of which nodes will come into communication range of each other after deployment.
- The computational and storage requirement of the scheme must be low, and the scheme should be robust to DoS attacks from out-of-network sources.

Evaluation metrics

Sensor networks have many characteristics that make them more vulnerable to attack than conventional computing equipment. Simply assessing a bootstrapping scheme based on its ability to provide secrecy is insufficient. Listed below are several criteria that represent desirable characteristics for a bootstrapping scheme for sensor networks.

- *Resilience against node capture*. It is assumed the adversary can mount a physical attack on a sensor node after it is deployed and read secret information from its memory. A scheme's resilience toward node capture is calculated by comparing the number of nodes captured, with the fraction of total network communications that are exposed to the adversary *not including* the communications in which the compromised nodes are directly involved.
- *Resistance against node replication*. Whether the adversary can insert additional hostile nodes into the network after obtaining some secret information (e.g., through node capture or infiltration). This is a serious attack since the compromise of even a single node might allow an adversary to populate the network with clones of the captured node to such an extent that legitimate nodes could be outnumbered and the adversary can thus gain full control of the network.
- *Revocation*. Whether a detected misbehaving node can be dynamically removed from the system.
- *Scalability*. As the number of nodes in the network grows, the security characteristics mentioned above may be weakened.

Each bootstrapping protocol usually involves several steps. An *initialization* procedure is performed to initialize sensor nodes before they are deployed. After the sensor nodes are deployed, a *key setup* procedure is performed by the nodes to set up shared secret keys between some of the neighboring nodes to establish a secure link.

4. Using a Single Network-Wide Key

The simplest method of key distribution is to pre-load a single network-wide key onto all nodes before deployment. After deployment, nodes establish

communications with any neighboring nodes that also possess the shared network key. This can be achieved simply by encrypting all communications in the shared network-wide key and appending a *message authentication code* (MAC) to ensure integrity. The Achilles heel of a single network-wide key is that a single node compromise results in a complete network compromise, and as sensor nodes are usually not robust to tampering, this is a significant concern.

The properties of the single network-wide key approach are as follows:

- *Minimal memory storage required*. Only a single cryptographic key is needed to be stored in memory.
- *No additional protocol steps are necessary*. The protocol works without needing to perform key discovery or key exchange. This represents savings in code size, node hardware complexity and communication energy costs.
- *Resistant against DoS, packet injection*. The MACs guard against arbitrary packet injection by an adversary that does not know the network-wide key k. Replay attacks can be prevented by including the source, destination, and a timestamp in the message. A worst case denial-of-service attack would be to replay large numbers of packets in order to force nodes to perform many MAC verifications. However, symmetric cryptographic MAC verifications are efficient and this kind of DoS attack would not be effective at preventing normal operation.

The main drawback of the network-wide key approach is that the compromise of a single node causes the compromise of the entire network, since the network-wide key is now known to the adversary. This makes it impractical except in two possible scenarios.

- *The nodes are tamper-resistant*. In this case, tamper resistance is incorporated into the sensor nodes such that it becomes challenging for an adversary to extract the network-wide key. In general, this is impractical for safety-critical sensor systems (such as security or safety applications) since adversaries attacking these systems will have a large incentive to defeat the tamper-resistance. Basagni et al. use this approach to design a secure routing protocol [3].
- *No new nodes are ever added to the system after deployment*. In this case, the sensor nodes use the network wide key to encrypt unique *link keys* which are exchanged with each of their neighbors. For example, node A might generate a unique key k_{AB} and send it to B encrypted with the network-wide key. Once the link keys are in place, all communications are encrypted using the appropriate link keys and the network-wide key is erased from the memory of the nodes. Any node which

is subsequently compromised thus reveals no secret information about the rest of the network. A major limitation of this approach is that no new nodes can later be added to the network, which is usually necessary for replacing failed nodes or expanding the sensor network. A possible way to address this would be to perform a large-scale audit of all sensor nodes prior to every phase of adding new nodes. The audit would have to ensure that every node's hardware and software has not been altered maliciously. Once this has been ascertained, a new network-wide key could be distributed to the nodes to enable addition of the new nodes. However, such a comprehensive audit would probably be too costly to be practical in most applications. Zhu, Setia, and Jajodia follow this approach and set up all keys from a single network-wide key during a short, initial phase after deployment, assuming that no nodes are compromised during this phase, and later all nodes erase the single network key [32]. This approach, however, is vulnerable to compromise of a single node that misses the key setup period, and does not erase its key.

5. Using Asymmetric Cryptography

The favored method of key distribution in most modern computer systems is via asymmetric cryptography, also known as public key methods. If sensor node hardware is able to support the computationally intensive asymmetric cryptographic operations, then this is a potentially viable method of key distribution.

A brief outline of a possible public-key method for sensor networks is as follows. Prior to deployment, a master public/private keypair, (K_M, K_M^{-1}) is first generated. Then, for every node A, its public/private keypair (K_A, K_A^{-1}) is generated. This keypair is stored in node A's memory along with the master public key K_M and the master key's signature on A's public key. Once all the nodes are initialized in this fashion, they are ready for deployment.

Once the nodes have been deployed, they perform key exchange. Nodes exchange their respective public keys and master key signatures. Each node's public key is verified as legitimate by verifying the master key's signature using the master public key. Once the public key of a node has been received, a symmetric link key can be generated and sent to it, encrypted by its public key. Upon reception of the session key, key establishment is complete and the two nodes can communicate using the symmetric link key. A smorgasbord of protocols exist for this purpose, we refer to standard cryptography text books for their description [20, 27]. Carman, Kruus and Matt present an analysis of the efficiency of several asymmetric key distribution protocols on various sensor nodes [9].

The properties of this approach are as follows:

- *Perfectly resilient against node capture*. Capture of any number of nodes does not expose any additional communications in the network, since these nodes will have no knowledge of any secret link keys besides the ones that they are actively using.

- *Possible to revoke known compromised keypairs*. Revocation can be performed by broadcasting a revocation message, signed by the master key. Nodes receiving the broadcast can authenticate it as coming from the central authority and ignore any future communications purporting to originate from the revoked keypair. If a large number of keypairs are to be revoked, the master keypair itself could be updated by broadcasting an updated new master key signed by the old master key, then unicasting the new master key's signature on each of the legitimate public keys (the unicast is encrypted in the respective nodes' public keys). The revoked keypairs will not be signed by the new master key and will thus be rejected as invalid by all nodes in the network.

- *Fully scalable*. Signature schemes function just as effectively regardless of the number of nodes in the network.

However, using asymmetric cryptography has its disadvantages, as follows:

- *Dependence on asymmetric key cryptographic hardware or software*. All known asymmetric cryptographic schemes often involve computationally intensive mathematical functions, such as the modular exponentiation of large numbers. Basic sensor node CPU hardware, however, is extremely limited, some even lack an integer multiply instruction. In order to implement asymmetric cryptography on sensor nodes, it is necessary to either incorporate dedicated cryptographic hardware on a sensor node, thus increasing its hardware cost, or encode the mathematical functions in software, which is usually $3 - -5$ orders of magnitude slower than symmetric cryptographic functions. Given that asymmetric cryptography is only used for key setup upon node deployment, this represents a tiny fraction of the sensor node's lifetime. Significantly increasing the cost of a node is thus difficult to justify.

- *Vulnerability to denial-of-service*. Asymmetric cryptographic operations involve significant amounts of computation and it can take from a few seconds up to minutes of processing for a sensor node to complete one signature generation or verification. Thus, the nodes are vulnerable to a battery exhaustion denial of service attack where they are continuously flooded with illegal signatures, or requested to generate a signature. Since this denial-of-service can only occur during the key establishment phase, the sensor network is only vulnerable when it is newly

deployed or when additional nodes are being added to the network. Increased monitoring of the site for adversarial transmissions during those times could alleviate the DoS problem.

- *No resistance against node replication*. Once a node is captured, its keypair can be used to set up links with every single one of the nodes in the network, effectively making the node "omnipresent". This could potentially put it in a position to subvert the routing infrastructure. We could prevent such an attack with some added countermeasures. For example, each time a node forms a connection with another node, they could both report the event to a base station encrypted using their master public key. Any node that has an exceptionally high degree could then be immediately revoked.

6. Using Pairwise-shared Keys

In this approach, every node in the sensor network shares a unique symmetric key with every other node in the network. Hence, in a network of n nodes, there are a total of $\binom{n}{2}$ unique keys. Every node stores $n-1$ keys, one for each of the other nodes in the network.

After deployment, nodes must perform key discovery to verify the identity of the node that they are communicating with. This can be accomplished with a challenge/response protocol.

The properties of this approach are as follows:

- *Perfect resilience to node capture*. Similar to the asymmetric cryptography scheme, any node that is captured reveals no information about the communications being performed in any other part of the network. Its pairwise keys could be used to perform a node replication attack throughout the network, but this could be countered using the same method as described for asymmetric cryptography in the previous section.

- *Compromised keys can be revoked*. If a node is detected to be compromised, its entire set of $n-1$ pairwise keys is simply broadcast to the network. No authentication is necessary. Any node that hears a key in its set of pairwise keys broadcast in the open immediately stops using it. This effectively cuts off the revoked node from the network.

- *Only uses symmetric cryptography*. The pairwise keys scheme achieves many of the benefits of using asymmetric cryptography without needing dedicated hardware or software to before the more complex asymmetric cryptographic primitives. This not only makes the nodes cheaper but also makes the network less vulnerable to denial of service attacks.

The main problem with the pairwise keys scheme is poor scalability. The number of keys that must be stored in each node is proportional to the total number of nodes in the network. With an 80 bit key, a network with 100 nodes will require almost 1kB of storage on each node for keys alone. Assuming memory-constrained sensor nodes, the pairwise keys scheme would not scale to large sensor networks. In addition, adding new nodes may also be a challenge in this setting.

7. Bootstrapping Security off a Trusted Base Station

This method of key distribution uses a trusted, secure base station as an arbiter to provide link keys to sensor nodes, e.g., similar to Kerberos [22, 15]. The sensor nodes authenticate themselves to the base station, after which the base station generates a link key and sends it securely to both parties. An example of such a protocol is part of the SPINS security infrastructure [24].

A sketch of the events of the protocol could be as follows. Prior to deployment, a unique symmetric key is generated for each node in the network. This node key is stored in the node's memory and will serve as the authenticator for the node as well as facilitate encrypted communications between the node and the base station. The base station has access to all the node keys either directly (they are stored in its memory) or indirectly (the base station relays all communications to a secured workstation off site).

This method, unlike the other methods mentioned previously, assumes some level of reliable transport is available between the node and the base station before any key establishment has taken place. Since this transport occurs before any security primitives are in place, it will necessarily have to be assumed as *insecure*, however, as long as it is *reliable* in a way such that a small number of malicious nodes are unable to prevent the transmission of messages to and from the base station then the protocol presented here is viable. *Flooding* (where nodes naively re-broadcast any heard messages until the entire network is reached) is a simple example of such a reliable method of transport, however this approach does not scale to large networks.

Now assume that after deployment, node A wants to establish a shared secret session key SK_{AB} with node B. Since A and B do not share any secrets, they need to use a trusted third party S, which is the base station in our case. SPINS includes a protocol where A and B can establish SK_{AB} through the base station [24]. The properties of this method of key establishment are as follows.

- *Small memory requirement.* For every node, a single secret symmetric key shared with the base station is needed, as well as one unique link key for each one of its neighbors. This is a small set of keys. Further-

more, the key establishment is efficient, as it only requires symmetric cryptographic primitives.

- *Perfect resilience to node capture*. Any node that is captured divulges no secret information about the rest of the network.

- *Revocation of nodes is simple*. Since no link keys can be established without the direct involvement of the base station, the base station has a record of all nodes that have established a link key with any given node. If a node is to be revoked, the base station securely transmits the revocation message to all the nodes that may be in communication with the revoked node. This revocation message is encrypted with the secret key that is shared only between the recipient node and the base station, and hence secrecy and authentication are ensured. To prevent any other nodes from establishing links with the revoked node, the base station simply needs to reject requests that involve the revoked node as a principal.

- *Node replication is easily controlled*. Since all key establishment activity takes place through the central base station, auditing becomes trivial. For example, the base station can immediately tell the current degree (number of established secure links) of any node. If this number is too high, then it refuses to generate any new link keys for that node. Thus node replication is effectively restricted.

However, key establishment through a base station has its disadvantages, as follows.

- *Not scalable—significant communication overhead*. If any two nodes wish to establish a secure communications, they must first communicate directly with the base station. In a large network, the base station may be many hops away, thus incurring a significant cost in communication. Worse still, since all large number of communications has the same source or destination (i.e., the base station), this may lead to contention and congestion at the nodes closest to the base station. The problem is further exacerbated by the fact that the transport must be able to function reliably without having any security primitives to rely on. This generally means that multiple transmission paths have to be used to prevent a malicious node from blocking a transmission. The communication overhead is thus further increased. Alternatively, multiple base stations may be used.

- *The base station becomes a target for compromise*. Since the base station has access to all the secret node keys in the sensor network, compromise

of the base station's key store will expose the secrecy of all links that are established after the time of the compromise. This may not be a problem if the communications base station merely acts as a gateway to a workstation at a remote, secured site, since the adversaries would have to successfully attack the secure workstation in order to gain the node keys. However, such a set up is not feasible for all applications, particularly sensor networks deployed far afield, for example, for seismic or wildlife monitoring. In such a situation the node keys have to be stored on the remote base station itself, which exposes them to attack.

8. Random Key Predistribution—Notation

The rest of the chapter focuses on one particular method of key distribution known as random key predistribution. For clarity, we list the symbols used in the following sections in the table below:

c	desired confidence level (probability) that the sensor network is connected after completing the connection protocol.
d	the expected degree of a node – i.e., the expected number of secure links a node can establish during key-setup.
m	number of keys in a node's key ring
n	network size, in nodes
n'	the expected number of neighbor nodes within communication radius of a given node
p	probability that two neighbor nodes can set up a secure link during the key-setup phase.
q	for the q-composite scheme, required amount of key overlap
S	key pool (set of keys randomly chosen from the total key space)
$\|S\|$	size of the key pool.
t	threshold number of votes after which a node will be revoked.

9. The Basic Random Key Predistribution Scheme

Eschenauer and Gligor first proposed random key-predistribution [12]. In the remainder of this chapter, we refer to their approach as the *basic scheme*. Let m denote the number of distinct cryptographic keys that can be stored on a sensor node. The basic scheme works as follows. Before sensor nodes are deployed, an *initialization phase* is performed. In the initialization phase, the basic scheme picks a random pool (set) of keys S out of the total possible key space. For each node, m keys are randomly selected from the key pool S and stored into the node's memory. This set of m keys is called the node's *key ring*. The number of keys in the key pool, $|S|$, is chosen such that two random subsets of size m in S will share at least one key with some probability p.

After the sensor nodes are deployed, a ***key-setup phase*** is performed. The nodes first perform key-discovery to find out with which of their neighbors they share a key. Such key discovery can be performed by assigning a short identifier to each key prior to deployment, and having each node broadcast its set of identifiers. Alternatively, the set of keys in the key ring of a node could be related to its ID via a pseudorandom function. In this case, each node need only broadcast its ID to its neighbors. While these broadcast-based key discovery methods are straightforward to implement, they have the disadvantage that a casual eavesdropper can identify the key sets of all the nodes in a network and thus pick an optimal set of nodes to compromise in order to discover a large subset of the key pool S. A more secure, but slower, method of key discovery could utilize client puzzles such as a Merkle puzzle [21]. Each node could issue m client puzzles (one for each of the m keys) to each neighboring node. Any node that responds with the correct answer to the client puzzle is thus identified as knowing the associated key.

Nodes which discover that they contain a shared key in their key rings then verify that their neighbor actually holds the key through a challenge-response protocol. The shared key then becomes the key for that link.

After key-setup is complete, a connected graph of secure links is formed. In some applications, this infrastructure is sufficient for normal operation. For example, if the communications in the network consists of mostly node-to-base or base-to-node communications, this infrastructure might be adequate assuming the base station can form secure links with *all* its neighboring nodes. However, if frequent node-to-node communications are performed, then the network needs each node to be able to form a secure link with all of its neighbors to improve long-term communication efficiency. In this case, nodes can set up *path keys* with nodes in their vicinity whom they did not happen to share keys with in their key rings. If the graph is connected, a path can be found from a source node to its neighbor. The source node can then generate a path key and send it securely via the path to the target node.

One needs to pick the right parameters such that the graph generated during the key-setup phase is connected. Consider a random graph $G(n, p_l)$, a graph of n nodes for which the probability that a link exists between any two nodes is p_l. Erdös and Rényi showed that for monotone properties of a graph $G(n, p_l)$, there exists a value of p_l over which the property exhibits a "phase transition", i.e., it abruptly transitions from "likely false" to "likely true" [25]. Hence, it is possible to calculate some expected degree d for the vertices in the graph such that the graph is connected with some high probability c, where $c = 0.999$, for example. Eschenauer and Gligor calculate the necessary expected node degree

d in terms of the size of the network n as:

$$d = \left(\frac{n-1}{n}\right)(\ln(n) - \ln(-\ln(c))) \tag{1.1}$$

From the formula, $d = O(\log n)$. In our examples we expect d to be in the range of 20 to 50.

For a given density of sensor network deployment, let n' be the expected number of neighbors within communication range of a node. Since the expected node degree must be at least d as calculated, the required probability p of successfully performing key-setup with some neighbor is:

$$p = \frac{d}{n'} \tag{1.2}$$

Since the models of connectivity are probabilistic, there is always the slight chance that the graph may not be fully connected. This chance is increased if the deployment pattern is irregular or the deployment area has unpredictable physical obstacles to communication. It is difficult to anticipate such scenarios prior to knowing the specifics of the deployment area. To address this, if the network detects that it is disconnected, sensor nodes may perform *range extension*. This may involve increasing their transmission power, or sending a request to their neighbors to forward their communications for a certain number of hops. Range extension may be gradually increased until a connected graph is formed after key-setup. A useful way for a node to detect if a network is connected is by checking if it can perform multi-hop communication with all base stations. If not, range extension should be performed.

10. The q-Composite Random Key Predistribution Scheme

In the basic scheme, any two neighboring nodes need to find a single common key from their key rings to establish a secure link in the key-setup phase. The q-composite keys scheme is a modification to the basic scheme where q common keys $(q > 1)$ are needed, instead of just one. By increasing the amount of key overlap required for key-setup, the scheme increases the resilience of the network against node capture.

Figure 1.10 reflects the motivation for the q-composite keys scheme. As the amount of required key overlap increases, it becomes exponentially harder for an attacker with a given key set to break a link. However, to preserve the given probability p of two nodes sharing sufficient keys to establish a secure link, it is necessary to reduce the size of the key pool $|S|$. This allows the attacker to gain a larger sample of S by breaking fewer nodes. The interplay of these two opposing factors results in an optimal amount of key overlap to pose the

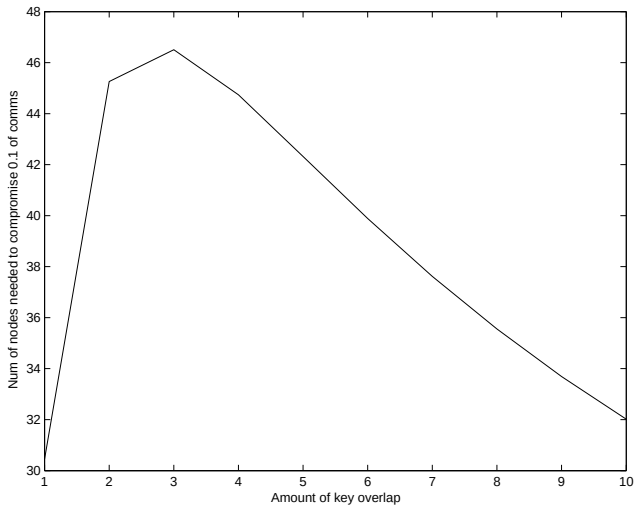


Figure 1.1. The expected number of nodes an adversary needs to capture before it is able to eavesdrop on any link with probability 0.1, for various amounts of key overlap q. Key ring size $m = 200$ keys, probability of connection $p = 0.5$.

greatest obstacle to an attacker for some desired probability of eavesdropping on a link.

Description of the q-Composite Keys Scheme

Initialization and Key Setup. The operation of the q-composite keys scheme is very similar to that of the basic scheme, differing only in the size of the key pool S and the fact that multiple keys are used to establish communications instead of just one.

The q-composite keys scheme first proceeds in a similar manner to the basic random keys scheme. That is, the deployer picks a set S of random keys out of the total key space, and for each node in the network, selects m random keys from S and stores them into the node's key ring. Nodes then perform key discovery with their neighbors.

After key discovery, each node can identify every neighbor node with which it shares at least q keys. Let the number of actual keys shared be q', where $q' \geq q$. A new communication link key K is generated as the hash of *all* shared keys, e.g., $K = hash(k_1||k_2||\ldots||k_{q'})$. The keys are hashed in some canonical order, for example, based on the order they occur in the original key pool S. Key-setup is not performed between nodes that share fewer than q keys.

Computation of Key Pool Size. We assume that we are required to take the sensor network's physical characteristics as a given parameter. Specifically, we are provided with a probability of full network connectivity c and the expected number of neighbors of each node n'. Via Equation 1.1, we first calculate d, the expected degree of any given node. This can be input to Equation 1.2 to calculate p, the desired probability that any two nodes can perform key-setup.

We now need to calculate the critical parameter $|S|$, the size of the key pool. If the key pool size is too large, then the probability of any two nodes sharing at least q keys would be less than p, and the network may not be connected after bootstrapping is complete. If the key pool size is too small, then we are unnecessarily sacrificing security. We would like to choose a key pool size such that the probability of any two nodes sharing at least q keys is $\geq p$. Let m be the number of keys that any node can hold in its key ring. We would like to find the largest S such that any two random samples of size m from S have at least q elements in common, with a probability of at least p.

We compute $|S|$ as follows. Let $p(i)$ be the probability that any two nodes have exactly i keys in common. Any given node has $\binom{|S|}{m}$ different ways of picking its m keys from the key pool of size $|S|$. Hence, the total number of ways for both nodes to pick m keys each is $\binom{|S|}{m}^2$. Suppose the two nodes have i keys in common. There are $\binom{|S|}{i}$ ways to pick the i common keys. After the i common keys have been picked, there remain $2(m-i)$ distinct keys in the two key rings that have to be picked from the remaining pool of $|S|-i$ keys. The number of ways to do this is $\binom{|S|-i}{2(m-i)}$. The $2(m-i)$ distinct keys must then be partitioned between the two nodes equally. The number of such equal partitions is $\binom{2(m-i)}{m-i}$. Hence the total number of ways to choose two key rings with i keys in common is the product of the aforementioned terms, i.e., $\binom{|S|}{i}\binom{|S|-i}{2(m-i)}\binom{2(m-i)}{m-i}$. Hence, we have

$$p(i) = \frac{\binom{|S|}{i}\binom{|S|-i}{2(m-i)}\binom{2(m-i)}{m-i}}{\binom{|S|}{m}^2} \tag{1.3}$$

Let $p_{connect}$ be the probability of any two nodes sharing sufficient keys to form a secure connection. $p_{connect} = 1-$ (probability that the two nodes share insufficient keys to form a connection), hence

$$p_{connect} = 1 - (p(0) + p(1) + \cdots + p(q-1)) \tag{1.4}$$

For a given key ring size m, minimum key overlap q, and minimum connection probability p, we choose the largest $|S|$ such that $p_{connect} \geq p$.

Evaluation of the q-Composite Random Key Distribution Scheme

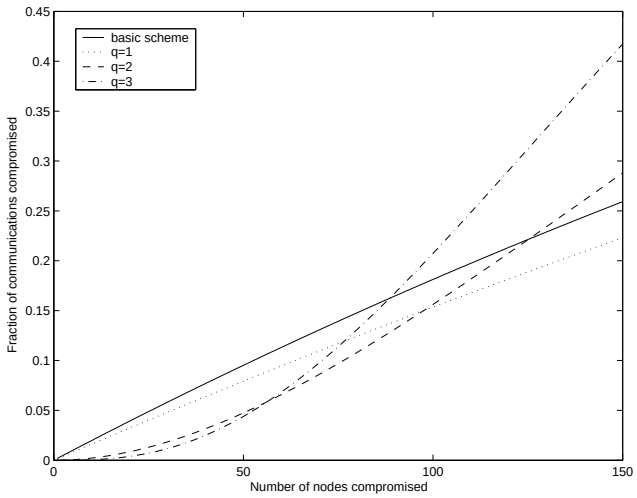


Figure 1.2. Probability that a specific random communication link between two random nodes A, B can be decrypted by the adversary when the adversary has captured some set of x nodes that does not include A or B. Key ring size $m = 200$, probability of key-setup $p = 0.33$.

Resilience Against Node Capture in q-Composite Keys Schemes.

Figure 1.2 shows what fraction of the communications in the remainder of the network is exposed with the number of nodes captured by the attacker.

Note that the scale of the x-axis shows absolute numbers of nodes compromised (i.e., independent of the actual total size of the network) while the y-axis is the fraction of the total network communications compromised. Hence, the schemes are not infinitely scalable - a compromise of x number of nodes will always reveal a fixed fraction of the total communications in the network regardless of network size.

The q-composite keys scheme offers greater resilience against node capture when the number of nodes captured is small. For example, in Figure 1.2a, for $q = 2$, the amount of additional communications compromised when 50 nodes have been compromised is 4.74%, as opposed to 9.52% for the basic scheme. However, when large numbers of nodes have been compromised, the q-composite keys schemes tend to reveal larger fractions of the network to the adversary. Increasing q makes it harder for an adversary to obtain small amounts of initial information from the network via a small number of initial node captures. This comes at the cost of making the network more vulnerable once a large number of nodes have been breached. This may be a desirable trade-off because small scale attacks are cheaper to mount and much harder to detect than large scale attacks. It is easy to mask an attack on a single node as a communications breakdown due to occlusion or interference; it is much harder to disguise an attack on many nodes as a natural occurrence.

The q-composite scheme removes the incentive for small scale attacks since the amount of additional information revealed in the rest of the network is greatly reduced. It forces the attacker to attempt large scale attacks which are expensive and more easily detectable.

11. Multipath Key Reinforcement

Multipath key reinforcement is a method to strengthen the security of an established link key by establishing the link key through multiple paths. This method can be applied in conjunction with the basic random key scheme to yield greatly improved resilience against node capture attacks by trading off some network communication overhead.

Description of Multipath Key Reinforcement

The basic idea behind multipath key reinforcement was first explored by Anderson and Perrig [1]. Assume that initial key-setup has been completed (in the following examples, we assume the basic random key scheme was used for key-setup). There are now many secure links formed through the common keys in the various nodes' key rings. Suppose A has a secure link to B after key-setup. This link is secured using a single key k from the key pool S. k may be residing in the key ring memory of some other nodes elsewhere in the network. If any of those nodes are captured, the security of the link between A and B is jeopardized. To address this, we would like to update the communication key to a random value after key-setup. However, we cannot simply coordinate the key update using the direct link between A and B since if the adversary has been recording all key-setup traffic, it could decrypt the key-update message after it obtained k and still obtain the new communication key.

The idea behind multipath key reinforcement is to coordinate the key-update over multiple independent paths. Assume that enough routing information can be exchanged such that A knows all disjoint paths to B created during initial key-setup that are h hops or less. Specifically, $A, N_1, N_2, \ldots, N_i, B$ is a path created during the initial key-setup if and only if each link (A, N_1), (N_1, N_2), $\ldots$, (N_{i-1}, N_i), (N_i, B) has established a link key during the initial key-setup using the common keys in the nodes' key rings. Let j be the number of such paths that are *disjoint* (do not have any links in common). A then generates j random values $v_1, \ldots, v_j$. Each random value has the same length as the encryption/decryption key. A then routes each random value along a different path to B. When B has received all j keys, then the new link key can be computed by both A and B as:

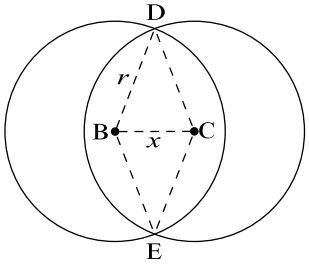
$$k' = k \oplus v_1 \oplus v_2 \oplus \ldots \oplus v_j$$

The secrecy of the link key k is protected by all j random values. Unless the adversary successfully manages to eavesdrop on all j paths, they will not know sufficient parts of the link key to reconstruct it.

The more paths that can be found between two nodes A and B, the more security multipath key reinforcement provides for the link between A and B. However, for any given path, the probability that the adversary can eavesdrop on the path increases with the length of the path since if any one link on the path is insecure then the entire path is made insecure. Further, it is increasingly expensive in terms of communication overhead to find multiple disjoint paths that are very long. It is instructive to analyze the case where only paths of 2 links (only one intermediate node) are considered. We call this scheme the *2-hop multipath key reinforcement scheme*. This approach has the advantage that path discovery overhead is minimized: for example, A could exchange neighbor lists with B. Once they identify their common neighbors with which both of them share a key, A and B can perform key reinforcement using their secure links through these common neighbors. Furthermore, the paths are naturally disjoint and no further effort needs to be taken to guarantee this property. We will calculate the expected effectiveness of this scheme and evaluate its security properties in simulation.

Estimation of Expected Effectiveness of 2-Hop Multipath Key Reinforcement

In this section, we first calculate the expected number of common neighbors between two nodes in a random uniform planar deployment of sensors. We then derive a formula for the new expected probability for compromising a given link after multipath key reinforcement has taken place.



The figure above indicates the parameters to be used in our calculation. B and C denote two communicating sensor nodes. r is the communications range of each sensor node. We assume that each node has the same range for receiving and transmitting. x is the distance between two nodes.

For any given separation x, the area $A(x)$ within both nodes' communication radii is the area of the sectors BDE and CDE minus the area of the rhombus $BDCE$:

$$A(x) = 2r^2 \cos^{-1}\left(\frac{x}{2r}\right) - x\sqrt{r^2 - \frac{x^2}{4}}$$

The probability distribution function of the distance between two nodes within communication radius is given by $F(x) = P(distance < x) = x^2/r^2$. The probability density function is thus $f(x) = F'(x) = 2x/r^2$. The expected area of overlap is thus given by:

$$\begin{aligned} & \int_0^r A(x)f(x)dx \\ = & \int_0^r \left(2r^2 \cos^{-1}\left(\frac{x}{2r}\right) - x\sqrt{r^2 - \frac{x^2}{4}}\right) \frac{2x}{r^2}\, dx \\ = & \left(\pi - \frac{3\sqrt{3}}{4}\right) r^2 \;=\; 0.5865\pi r^2 \end{aligned}$$

We define the term *reinforcing neighbors* of two nodes sharing a secure link as the common neighbors with whom both nodes share a secure link. Since the expected area of overlap is 0.5865 of a single communication radius, the expected number of reinforcing neighbors is thus $0.5865p^2n'$ where p is the probability of sharing sufficient keys to communicate, and n' is the number of neighbors of each node. Via Equation 1.2, this can also be expressed as $0.5865\frac{d^2}{n'}$. As an example, for $d = 20$ and $n' = 60$ (i.e. $p = 0.33$), the expected number of reinforcing neighbors is 3.83.

In general, if a link is reinforced by k common neighbors, then the adversary must be able to eavesdrop on that link, as well as at least one link on each of the k 2-hop paths. If the adversary's base probability of compromising a link is b, then the probability of compromising at least one hop on any given 2-hop path is the probability of compromising hop 1 in the path plus the probability of compromising hop 2 in the path minus probability of compromising both hops in the path = $2b - b^2$. Hence, the final probability of breaking the link is now

$$b' = b(2b - b^2)^k$$

For example, if the adversary has a base 0.1 chance of eavesdropping on a given link before reinforcement, for a link reinforced by 3 neighbors, the chance of eavesdropping after reinforcement improves to 6.86×10^{-4}, or about 1 in 1,458.

From the expected number of reinforcing neighbors we can estimate the expected network communications overhead of the 2-hop multipath reinforcement scheme. Each reinforcing neighbor represents an extra 2-hop commu-

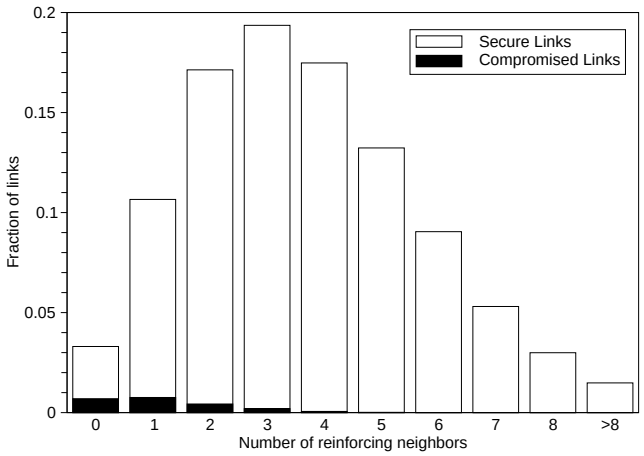


Figure 1.3. Reinforcement and compromise statistics for base compromise probability $b = 0.2$

nication to help reinforce a given 1-hop link. Hence, on average, the total additional communications overhead for key-reinforcement is at least $2 \times 0.5865p^2n'$ times more than the network communications needed for basic key-setup, not including additional communications for common-neighbor discovery. For example, for $p = 0.33$ and $n' = 60$, we can expect to see at least 7.66 times additional network traffic after key-setup is complete. Including common neighbor discovery, we estimate the final scheme to be approximately 10 times more expensive in network communications than the basic scheme in this case. Given that eavesdropping probabilities can be improved from 0.1 to 6.86×10^-4 (146 times improvement), this may be a good trade-off.

Evaluation of Multipath Key Reinforcement

The effectiveness of 2-hop multipath key reinforcement is evaluated here by simulating the random uniform deployment of 10,000 sensor nodes on a square planar field. The probability of any two nodes being able to establish a secure link is set at $p = 0.33$, and the deployment density is set such that the expected number of neighbors of each node was 60. The eavesdropping attack is modeled by iterating over each secure link and marking it as compromised with random chance based on the simulated probability of compromise c. A link is considered completely compromised only if it is compromised and all its reinforcement paths are also compromised.

Figure 1.3 reflects the relative distribution of the number of reinforcing neighbors for each link in the simulation. The results indicated reflect support for our calculated average of 3.83 reinforcing neighbors between any 2 nodes within communication distance. The figure also shows the distribution

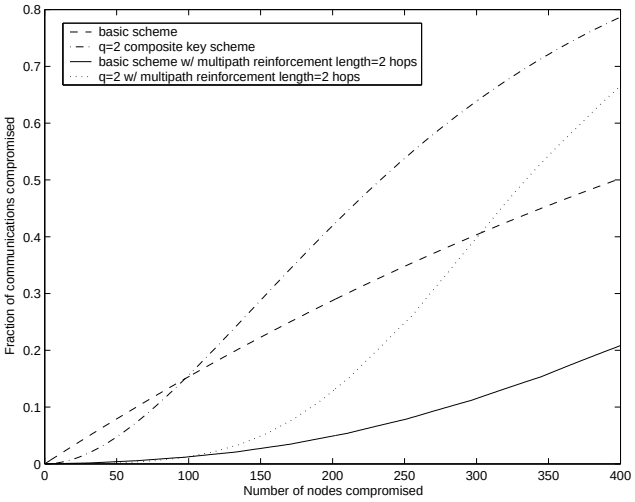


Figure 1.4. Multipath key reinforcement resistance against node capture ($m = 200, p = 0.33$)

of reinforced links that were compromised by an adversary with a base 0.2 probability of compromising any link prior to reinforcement. In this simulation, links with more than 3 reinforcing neighbors did not suffer significant rates of compromise. The overall rate of compromise was lowered by an order of magnitude, from 0.2 to 0.022.

Figure 1.4 indicates the amount of communications compromised versus the number of nodes compromised, with and without key reinforcement for the various schemes. Successfully implementing multipath key reinforcement on the basic scheme enables it to outperform the q-composite scheme for $q \geq 2$ even when the q-composite scheme is supplemented by key reinforcement. The intuitive reason for this is that multipath key reinforcement acts similarly to the q-composite keys scheme in that it compounds the difficulty of compromising a given link by requiring the adversary possess multiple relevant keys to eavesdrop on a given link. The trade-off for this benefit in the q-composite case is a smaller key pool size; the trade-off for the multipath key reinforcement scheme is increased network overhead. Compounding both the schemes compounds their weaknesses - the smaller key pool size of the q-composite keys scheme undermines the effectiveness of multipath key reinforcement by making it easier to build up a critically large collection of keys.

The cost of the improved security due to multipath key reinforcement is an added overhead in neighbor discovery and key establishment traffic. Whether this tradeoff is a good one will depend on the specific application as well as the deployment density characteristics of the sensor network.

While the analysis presented is for using multipath key reinforcement to secure links that have been formed after key-setup, the scheme can also be used to reinforce *path-keys* that are established between nodes that did not share keys during key setup. This will further improve the security of the system.

12. Pairwise Key Schemes

In the random key pool distribution schemes described above, keys can be issued multiple times out of the key pool, and node-to-node authentication is not possible [10]. In contrast, pairwise key distribution assigns a unique key to each pair of nodes. In this section, we review several different approaches for pairwise key distribution: the random pairwise key scheme by Chan, Perrig and Song [10], the single-space pairwise key distribution approaches by Blom [6] and Blundo et al. [7], and the multi-space pairwise key scheme by Du et al. [11] and by Liu and Ning [18].

Random Pairwise Key Scheme

Recall that the size of each node's key rings is m keys, and the probability of any two nodes being able to communicate securely is p. The random pairwise keys scheme proceeds as follows:

1 In the pre-deployment *initialization* phase, a total of $n = \frac{m}{p}$ unique node identities are generated. The actual size of the network may be smaller than n. Unused node identities will be used if additional nodes are added to the network in the future. Each node identity is matched up with m other randomly selected distinct node IDs and a pairwise key is generated for each pair of nodes. The key is stored in both nodes' key rings, along with the ID of the other node that also knows the key.

2 In the post-deployment *key-setup* phase, each node first broadcasts its node ID to its immediate neighbors. By searching for each other's IDs in their key-rings, the neighboring nodes can tell if they share a common pairwise key for communication. A cryptographic handshake is then performed between neighbor nodes who wish to mutually verify that they do indeed have knowledge of the key.

Single-Space Pairwise Key Schemes

Both Blom's and the polynomial scheme require a sensor node i to store unique public information U_i and private information V_i. During the bootstrapping phase, nodes exchange public information, and node i could compute its key with node j with $f(V_i, U_j)$. It is guaranteed that $f(V_i, U_j) = f(V_j, U_i)$. Both approaches ensure the λ-secure property: the coalition of no more than

λ compromised sensor nodes reveals nothing about the pairwise key between any two non-compromised nodes.

Multi-Space Pairwise Key Schemes

Recently, researchers have proposed the idea of multiple key spaces [11, 18] to significantly enhance the security of single-space approaches. The idea of introducing multiple key spaces can be viewed as the combination of the basic key pool scheme and the above single space approaches. The setup server randomly generates a pool of m key spaces each of which has unique private information. Each sensor node will be assigned k out of the m key spaces. If two neighboring nodes have one or more key spaces in common, they can compute their pairwise secret key using the corresponding single space scheme.

13. Other Related Work

We first review work in establishing shared keys in mobile computing, then review work in sensor network key establishment.

Tatebayashi, Matsuzaki, and Newman consider key distribution for resource-starved devices in a mobile environment [28]. Leighton and Micali present two mechanisms for key agreement using a predistributed set of symmetric keys [17]. Their first scheme is similar in nature to our q-composite protocols, their keys in the key pool are deterministically selected, such that any two nodes can certainly establish a shared key. Park et al. [23] point out weaknesses and improvements. Beller and Yacobi further develop key agreement and authentication protocols [4]. Boyd and Mathuria survey the previous work on key distribution and authentication for resource-starved devices in mobile environments [8]. The majority of these approaches rely on asymmetric cryptography. Bergstrom, Driscoll, and Kimball consider the problem of secure remote control of resource-starved devices in a home [5].

Stajano and Anderson discuss the issues of bootstrapping security devices [26]. Their solution requires physical contact of the new device with a master device to imprint the trusted and secret information.

Carman, Kruus, and Matt analyze a wide variety of approaches for key agreement and key distribution in sensor networks [9]. They analyze the overhead of these protocols on a variety of hardware platforms.

Wong and Chan propose a key exchange for low-power devices [29]. However, their approach assumes an asymmetry in computation power, that is, one of the participants is a more powerful server.

Perrig et al. propose SPINS, a security architecture specifically designed for sensor networks [24]. In SPINS, each sensor node shares a secret key with the base station. To establish a new key, two nodes use the base station as a trusted third party to set up the new key.

We review the related work by Eschenauer and Gligor [12] in Section 1.9. Anderson and Perrig propose a key establishment mechanism for sensor networks based on initially exchanging keys in the clear [1]. Their key infection approach is secure as long as an attacker arrives after key exchange and did not eavesdrop on the exchange.

Zhou and Haas propose to secure ad hoc networks using asymmetric cryptography [31]. Kong et al. propose localized public-key infrastructure mechanisms, based on secret sharing and multiparty computation techniques [16]. Such approaches are expensive in terms of computation and communication overhead.

Broadcast encryption by Fiat and Naor [13] is another model for distributing a shared key to a group of receivers. However, this model assumes a single sender, and that the sender knows the key pools of all receivers. Subsequent papers further develop this approach [2, 14, 19].

References

[1] Ross Anderson and Adrian Perrig. Key infection: Smart trust for smart dust. Unpublished Manuscript, November 2001.

[2] Dirk Balfanz, Drew Dean, Matt Franklin, Sara Miner, and Jessica Staddon. Self-healing key distribution with revocation. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 241–257, May 2002.

[3] Stefano Basagni, Kris Herrin, Emilia Rosti, and Danilo Bruschi. Secure pebblenets. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 156–163, October 2001.

[4] M. Beller and Y. Yacobi. Fully-fledged two-way public key authentication and key agreement for low-cost terminals. *Electronics Letters*, 29(11):999–1001, May 1993.

[5] Peter Bergstrom, Kevin Driscoll, and John Kimball. Making home automation communications secure. *IEEE Computer*, 34(10):50–56, Oct 2001.

[6] R. Blom. Non-public key distribution. In *Advances in Cryptology: Proceedings of Crypto '82*, pages 231–236, 1982.

[7] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology - Crypto '92*, pages 471–486, 1992.

[8] Colin Boyd and Anish Mathuria. Key establishment protocols for secure mobile communications: A selective survey. In *Australasian Conference on Information Security and Privacy*, pages 344–355, 1998.

[9] David W. Carman, Peter S. Kruus, and Brian J. Matt. Constraints and approaches for distributed sensor network security. *NAI Labs Technical Report #00-010*, September 2000.

[10] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, May 2003.

[11] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM CCS 2003*, pages 42–51, October 2003.

[12] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communication Security*, pages 41–47, November 2002.

[13] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, 1994.

[14] J. Garay, J. Staddon, and A. Wool. Long-lived broadcast encryption. In *Advances in Cryptology — CRYPTO '2000*, pages 333–352, 2000.

[15] John Kohl and B. Clifford Neuman. The Kerberos Network Authentication Service (V5). RFC 1510, September 1993.

[16] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *9th International Conference on Network Protocols (ICNP'01)*, 2001.

[17] T. Leighton and S. Micali. Secret-key agreement without public-key cryptography. In *Advances in Cryptology - Crypto '93*, pages 456–479, 1993.

[18] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *ACM CCS 2003*, pages 52–61, October 2003.

[19] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In *Advances in Cryptology — EUROCRYPT '98*, pages 512–526, 1998.

[20] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, 1997.

[21] R. Merkle. Secure communication over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[22] S. P. Miller, C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. In *Project Athena Technical Plan*, page section E.2.1, 1987.

[23] C. Park, K. Kurosawa, T. Okamoto, and S. Tsujii. On key distribution and authentication in mobile radio networks. In *Advances in Cryptology - EuroCrypt '93*, pages 461–465, 1993. Lecture Notes in Computer Science Volume 765.

[24] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001)*, July 2001.

[25] J. Spencer. *The Strange Logic of Random Graphs*. Number 22 in Algorithms and Combinatorics. 2000.

[26] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols, 7th International Workshop*, 1999.

[27] Douglas R. Stinson. *Cryptography: Theory and practice*. CRC Press, 2nd edition edition, 2002.

[28] M. Tatebayashi, N. Matsuzaki, and D. B. Jr. Newman. Key distribution protocol for digital mobile communication systems. In *Advances in Cryptology - Crypto '89*, pages 324–334, 1989. Lecture Notes in Computer Science Volume 435.

[29] Duncan S. Wong and Agnes H. Chan. Efficient and mutually authenticated key exchange for low power computing devices. In *Advances in Cryptology — ASIACRYPT '2001*, 2001.

[30] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pages 1–13, November 2003.

[31] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6):24–30, November/December 1999.

[32] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *ACM CCS 2003*, pages 62–72, October 2003.