Volume 30, issues 11–12, 10 September 2007 ISSN 0140-3664

computer communications

Available online at ScienceDirect www.sciencedirect.com

Special Issue:
Security on Wireless Ad Hoc and Sensor Networks
Guest Editors: Sghaier Guizani
Hsiao-Hwa Chen
Peter Mueller

Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2314–2341
computer communications
www.elsevier.com/locate/comcom

A survey of key management schemes in wireless sensor networks

Yang Xiao [a,*], Venkata Krishna Rayi [b], Bo Sun [c], Xiaojiang Du [d], Fei Hu [e], Michael Galloway [a]

[a] *Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487, USA*
[b] *XINOPT Co., 6421 brightlea dr, lanham, MD 20706, USA*
[c] *Department of Computer Science, Lamar University, Beaumont, TX 77710, USA*
[d] *Department of Computer Science, North Dakota State University, Fargo, ND 58105, USA*
[e] *Computer Engineering Department, Rochester Institute of Technology, Rochester, NY 14623, USA*

Available online 10 May 2007

Abstract

Wireless sensor networks have many applications, vary in size, and are deployed in a wide variety of areas. They are often deployed in potentially adverse or even hostile environment so that there are concerns on security issues in these networks. Sensor nodes used to form these networks are resource-constrained, which make security applications a challenging problem. Efficient key distribution and management mechanisms are needed besides lightweight ciphers. Many key establishment techniques have been designed to address the tradeoff between limited memory and security, but which scheme is the most effective is still debatable. In this paper, we provide a survey of key management schemes in wireless sensor networks. We notice that no key distribution technique is ideal to all the scenarios where sensor networks are used; therefore the techniques employed must depend upon the requirements of target applications and resources of each individual sensor network.

Keywords: Wireless sensor network; Key management; Security

1. Introduction

Wireless Sensor Networks (WSNs) are going to be widely used in the near future due to their breadth of applications by military, exploration teams, researchers, and so on. It is not possible to use general wireless techniques for WSNs since they are resource-constrained and security measures are required. Distribution techniques that are applicable employ assorted key management methods, such as public key cryptography, and require numerous communication and computation capabilities. Therefore, it is important to examine the different requirements, constraints and evaluation metrics of sensor networks as well as single network-wide key scheme, which is the simplest of key management techniques, before discussing the various germane key management techniques.

Sensor networks must arrange several types of data packets, including packets of routing protocols and packets of key management protocols. The key establishment technique employed in a given sensor network should meet several requirements to be efficient. These requirements may include supporting in-network processing and facilitating self-organization of data, among others. However, the key establishment technique for an secure application must minimally incorporate authenticity, confidentiality, integrity, scalability, and flexibility.

- *Authenticity*: The key establishment technique should guarantee that the communication nodes in the network have a way for verifying the authenticity of the other nodes involved in a communication, i.e., the receiver node should recognize the assigned ID of the sender node.

[*] Corresponding author.
E-mail addresses: yangxiao@ieee.org (Y. Xiao), Krishna.rayi@gmail.com (V.K. Rayi), bsun@cs.lamar.edu (B. Sun), dxj@ieee.org (X. Du), fxheec@rit.edu (F. Hu), mgalloway@cs.ua.edu (M. Galloway).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.04.009

- *Confidentiality*: The key establishment technique should protect the disclosure of data from unauthorized parties. An adversary may try to attack a sensor network by acquiring the secret keys to obtain data. A better key technique controls the compromised nodes to keep data from being further revealed.
- *Integrity*: Integrity means no data falsification during transmissions. Here in terms of key establishment techniques, the meanings are explained as follows. Only the nodes in the network should have access to the keys and only an assigned base station should have the privilege to change the keys. This would effectively prevent unauthorized nodes from obtaining knowledge about the keys used and preclude updates from external sources.
- *Scalability*: Efficiency demands that sensor networks utilize a scalable key establishment technique to allow for the variations in size typical of such a network. Key establishment techniques employed should provide high-security features for small networks, but also maintain these characteristics when applied to larger ones.
- *Flexibility*: Key establishment techniques should be able to function well in any kind of environments and support dynamic deployment of nodes, i.e., a key establishment technique should be useful in multiple applications and allow for adding nodes at any time.

One of the challenges in developing sensor networks is to provide high-security features with limited resources. Sensor networks cannot be costly made as there is always a great chance that they will be deployed in hostile environments and captured for key information or simply destroyed by an adversary, which, in turn, can cause huge losses. Part of these cost limitation constraints includes an inability to make sensor networks totally tamper-proof. Other sensor node constraints that must be kept in mind while developing a key establishment technique include battery life, transmission range, bandwidth, memory, and prior deployment knowledge.

- *Battery life*: Sensor nodes have a limited battery life, which can make using asymmetric key techniques, like public key cryptography, impractical as they use much more energy for their integral complex mathematical calculations. This constraint is mitigated by making use of more efficient symmetric techniques that involve fewer computational procedures and require less energy to function.
- *Transmission range*: Limited energy supply also restricts transmission range. Sensor nodes can only transmit messages up to specified short distances since increasing the range may lead to power drain. Techniques like in-network processing can help to achieve better performance by aggregating and transmitting only processed information by only a few nodes. This way save the dissipated energy.
- *Bandwidth*: It is not efficient to transfer large blocks of data with the limited bandwidth capacity of typical sensor nodes, such as the transmitter of the UC Berkeley Mica platform that only has a bandwidth of 10Kbps. To compensate, key establishment techniques should only allow small chunks of data to be transferred at a time.
- *Memory*: Memory availability of sensor nodes is usually 6–8 Kbps, half of which is occupied by a typical sensor network operating system, like TinyOS. Key establishment techniques must use the remaining limited storage space efficiently by storing keys in memory, buffering stored messages, etc.
- *Prior deployment knowledge*: As the nodes in sensor networks are deployed randomly and dynamically, it is not possible to maintain knowledge of every placement. A key establishment technique should not, therefore, be aware of where nodes are deployed when initializing keys in the network.

A key establishment technique is not judged solely based upon its ability to provide secrecy of transferred messages, but must also meet certain other criteria for efficiency in light of vulnerability to adversaries, including the three Rs of sensor networks: resistance, revocation, and resilience. Though scalability may be considered an evaluation metric, it is not discussed here since we have included it in WSN requirements.

- *Resistance*: An adversary might attack the network by compromising a few nodes in the network and then replicating those nodes back into the network. Using this attack the adversary can populate the whole network with his replicated nodes and thereby gain control of the entire network. A good key establishment technique must resist node replication to guard against such attacks.
- *Revocation*: If a sensor network become invaded by an adversary, the key establishment technique should provide an efficient way to revoke compromised nodes, a lightweight method that does not use much of the network's already limited capacity for communication.
- *Resilience*: If a node within a sensor network is captured, the key establishment technique should ensure that secret information about other nodes is not revealed. A scheme's resilience is calculated using the total number of nodes compromised and the total fraction of communications compromised in the network. Resilience also means conveniently making new inserted sensors to join secure communications.

We classify key management schemes [1–34] in wireless sensor networks as follows, while details of the schemes will be discussed in later sections: (1) Single network-wide key, (2) Pairwise key establishment, (3) Trusted base station, (4) Public key schemes (elliptic curve cryptography) [25–28], (5) Key predistribution schemes (random key predistribution scheme [1], q-Composite random key predistribution scheme [2], Multipath reinforcement scheme [2], Random pairwise key scheme [2], Polynomial pool-based key predistribution [3], Random subset key predistribution [3], Grid-based key predistribution [3], Hypercube key distribution scheme [31], Key management schemes using deployment knowledge [8], Location dependent key management scheme [32], Location aware combinatorial key management [33], etc.), (6) Dynamic key management [30], and (7) Hierarchical key managements (LEAP [6], Heterogeneous sensor networks [24,25]), etc.

The key establishment technique employed in a given sensor network should take into consideration all the requirements, constraints, and evaluation metrics discussed. In our work we have assessed different types of key establishment techniques, each ranging in efficiency by providing various necessary characteristics.

In Section 2 of this chapter we explain single network-wide key, pairwise key establishment, trusted base station, and authentication. Section 3 discusses public key schemes, in particular elliptic curve cryptography schemes. Section 4 presents various key predistribution schemes. In Section 5, we introduce dynamic key management schemes. We survey hierarchical key management schemes in Section 6. Finally, we conclude this paper in Section 7.

2. Single network-wide key, pairwise key establishment, trusted base station, authentication

In this section, we briefly introduce single network-wide key, pairwise key establishment, and trusted base station. Finally, we also introduce an authentication scheme since it is used by many schemes in the later sections.

2.1. Single network-wide key

Using a single network-wide key is by far the simplest key establishment technique. In the initialization phase of this technique, a single key is preloaded into all the nodes of the network. After deployment, every node in the network can use this key to encrypt and decrypt messages. Some of the advantages offered by this technique include minimal storage requirements and avoidance of complex protocols. Only a single key is to be stored in the nodes' memory and once deployed in the network, there is no need for a node to perform key discovery or key exchange since all the nodes in communication range can transfer messages using the key which they already share.

Though a single network-wide key may seem advantageous, the main drawback is that compromise of a single node causes the compromise of the entire network through the shared key. This scheme counters several constraints with less computation and reduced memory use, but it fails in providing the basic requirements of a sensor network by making it easy for an adversary trying to attack.

2.2. Pairwise key establishment scheme

The pairwise key establishment scheme, however, is one of the most efficient key establishment schemes in wireless sensor networks because it does offer many additional features compared to other schemes, including node-to-node authentication and resilience to node replication.

For a network of n nodes in the Pairwise Scheme, the key predistribution is done by assigning each node a unique pairwise key with all the other nodes in the network, i.e., $n-1$ pairwise keys, which are retained in each node's memory so that each node can communicate with all the nodes in its communication range. With each node sharing a unique key with every other node in the network, this scheme offers node-to-node authentication. Each node can verify the identity of the node it is communicating with. This scheme also offers increased resilience to network capture as a compromised node does not reveal information about other nodes that are not directly communicating with the captured node. Through increased resilience, the scheme minimizes the chance for node replication. The drawback with the Pairwise Scheme is the additional overhead needed for each node to establish $n-1$ unique keys with all the other nodes in the network and maintain those keys in its memory. Utilizing such a scheme makes a network size prohibitive since, as the number of nodes in the network increases, so do the number keys that must be stored in each node's memory. If there is a network of 10,000 nodes, then each node must store 9999 keys in their memory. Since sensor nodes are resource-constrained, this significant overhead limits the scheme's applicability, but it can be effectively used for smaller networks.

2.3. Trusted base station

The main problem of using pairwise key establishment scheme is that every node in the network has to store $n-1$ key pairs. This can be eradicated when we use a trusted base station to send the session keys for the communication between any two nodes. The scheme is also called centralized key distribution center (KDC) approach. The scheme has small memory requirement and perfectly controlled node replication, It is resilient to node capture and possible to revoke key pairs. The drawbacks are that it is not scalable and the base station becomes the target of attacks.

2.4. Authentication: μTESLA

Perrig, Szewczyk, Tygar, Wen and Culler [5] at UC Berkeley presented a suite of security protocols optimized for sensor networks that they called 'SPINS'. The suite is built upon two secure building blocks, each performing individual required work: SNEP and μTESLA. SNEP offers data confidentiality, authentication, integrity, and freshness, while μTESLA offers broadcast data authentication. The μTESLA protocol, used on regular networks, is modified as a SPINS for use in resource-constrained WSNs. SPINS incorporates TinyOS (operating system) in each node, all of which communicate with a base station. Most WSN communications pass through the base station and involve three communication types: node-to-base station, base station-to-node, and base station-to-all nodes.

The main goal of SPINS protocol is to design a key establishment technique based on SNEP and μTESLA to prevent an adversary from spreading to other nodes in the network through a compromised node. Each node in this scheme shares a secret key with the base station that is initialized before deployment. The following are some of the representations in this scheme used to illustrate how this works:

- Node A and node B are two communicating nodes in the network;
- N_a is generated by node A;
- X_{ab} is the master key shared between nodes A and B;
- K_{ab} and K_{ba} are the encryption keys shared between node A and node B, which are derived from the master key X_{ab};
- K'_{ab} and K'_{ba} are the secret *MAC* keys shared between node A and node B, which are derived from the master key X_{ab};
- $\{M\}K_{ab}$ denotes the encryption of message M with key K_{ab};
- $MAC(K'_{ab}, M)$ denotes the computation of *MAC* for message M with *MAC* key K'_{ab}.

(1) SNEP: Data confidentiality/authentication/freshness: A combination of two schemes forms SNEP including a counter for semantic security and a bootstrapping scheme. Using this combination, SNEP is able to offer a number of advantages and only adds 8 bytes per message by reducing the communication overhead of the network. It uses a counter, like many other protocols, to offer authentication and freshness, but does so using means that also provide *semantic security*. Note that semantic security is nothing new and it is a common technique in cryptography, such as using the traditional counter mode. Two counters are shared between nodes attempting to communicate with each other for which some of the source node's cryptographic techniques send the shared counters with a message to the destination node. General encryption can be used as a simple form of confidentiality, but is not sufficient to protect messages; whereas, *semantic security* offers far greater security by making it harder for an adversary to derive the original data even after obtaining one or more encrypted messages. In WSNs, sending messages with a counter can cause overhead; but, the energy can be saved by sharing the counter between both nodes and incrementing it each time the destination node receives a message. As with other schemes, for better security the same keys should not be used again and again. In SNEP, independent keys are used for encryption and *MAC* operations. The secret key shared between source node A and destination node B is used for deriving the encryption and *MAC* keys for each direction. The encrypted data has the form $E = D(K,C)$ where D is the data, K is the encryption key and C is the counter. The MAC is $M = MAC(K',C\|E)$.

In SNEP, the total message that node A sends to node B is: $A \rightarrow B : D(K_{ab}, C_a), MAC(K'_{ab}, C_a||D(K_{ab}, C_a))$.

The *semantic security* property is satisfied as each time the message is encrypted, the counter value is incremented to a different value; thus, though the same message is encrypted, an adversary would not be able to decode the message. This is exactly the same as the traditional counter mode in cryptography.

With SNEP, an adversary does have a chance of performing a DOS attack by constantly sending the requests for counter synchronization, but this can be prevented either by sending the counter value with each encrypted message or by attaching a short MAC to the message that does not depend on the counter. Data authentication is done using the *MAC*. The counter value in the message prevents an adversary from replaying old messages, which would cause confusion and overhead in a WSN. As the counter value is kept at both ends of communication and the ID is not transferred with every message, communication overhead is negligible. The counter scheme also allows achieving weak freshness. If the counter value is verified correctly, it reveals the sequence of the messages, but only guarantees the sequence of messages, not that the reply from node B is caused by the message from node A. To achieve strong freshness that includes delay estimation, a none must be included with messages. To achieve strong freshness, node A sends a nonce N_a along with a reply message to node B, which resends the nonce with a reply message. This process can be optimized by implicitly using the nonce in the *MAC* computation; therefore, the entire SNEP protocol with strong freshness is: $A \rightarrow B : N_a, R_a$ and $B \rightarrow A : \{R_b\}_{(K_{ba},C_b)}, MAC(K'_{ba}, N_a ||C_b||\{R_b\}_{(K_{ba},C_b)})$.

If the *MAC* correctly verifies, node A will know that the reply from B is a reply to its message. In this method, it is assumed that both communicating parties know the counter value so that it need not be sent with every message; though, in reality, messages might get lost or tampered and cause inconsistencies in the counter value. Protocols needed to synchronize the counter value include bootstrapping the counter value in the following manner: $A \rightarrow B : C_a$ with $B \rightarrow A : C_b, MAC(K'_{ba}, C_a||C_b)$ and $A \rightarrow B : MAC(K'_{ab}, C_a||C_b)$.

The counter value need not be encrypted since the protocol needs strong freshness for which both communicating parties use the counter as nonce. Also, *MAC* need not include the names A and B as the keys they use, K_{ab}, state which nodes are participating in the communication. If node A realizes that the counter C_b of node B is not synchronized, it may request the counter of B with a message including N_a for strong freshness, or $A \rightarrow B : N_a$ and $B \rightarrow A : C_b, MAC(K'_{ba}, N_a||C_b)$.

(2) μTESLA: Authenticated broadcasts: Authenticating broadcasted data is a critical issue in WSNs, but previous solutions to this problem suffer from too much communication and computation overhead, and therefore, are not so useful in resource-constrained WSNs. TESLA, one of these solutions, provides an inefficient scheme for broadcasting data with authentication by using the digital signatures technique, which adds 24 bytes of overhead to each message that are typically only allotted a packet size of 30 bytes. Thus, using TESLA can cause almost all of the packet size to be occupied for the code only. Also, TESLA discloses the key with every message packet it sends and receives, which can use a great deal of a WSN's energy. Finally, TESLA authenticates keys using a one-way key chain, which is not possible to be stored in each sensor node. Perrig et al. modified TESLA for authenticating broadcasted data in a way that involves no significant overhead, called μTESLA, this method reduces energy needed by authenticating data using asymmetric mechanisms. Also unlike TESLA, which discloses the key every time a packet is sent or received, μTESLA does so only once in an epoch. The only limit with μTESLA is that it restricts the number of authenticated senders as it is expensive to store the one-way key chain in a sensor node.

μTESLA is able to provide the asymmetric cryptographic type of authenticated broadcast through delayed disclosure of symmetric keys. For broadcasting authenticated information between the base station and nodes of a WSN, μTESLA requires that the base station and nodes are loosely time synchronized and that each node knows an upper bound on the maximum synchronization error. When the base station wants to sends a packet to all the nodes in a given network, it computes a MAC on the packet beforehand. Since all the nodes in the network are sure that only a base station can compute the MAC, the MAC key is not disclosed at this point in time so they will not be vulnerable to attacks from an adversary. The packets sent to the nodes are stored in their buffers until the base station discloses the corresponding keys. Once disclosed, the keys can be authenticated by the nodes' using the one-way function F. If a key is correct, a node can use it to authenticate the packet stored in its buffer.

Each MAC key is a sequence of keys generated by the function F. The sender chooses the last key K_n of the chain randomly and then generates the one-way key chain by repeatedly applying F. Supposing that the base station has sent packets P_1 and P_2 in the time interval t_1, P_3 and P_4 in t_2, P_5 in t_3, and P_6 in t_4, the nodes receiving the packets cannot verify their authentication immediately, so the nodes store them in buffers. Packets sent in a particular time interval are authenticated using the key that corresponds to that time interval. Let the difference of the time interval be two in this case, and the receiver node is loosely time synchronized with the base station and knows key K_0. Assuming that all the messages sending the key information about packets $P_1 - P_5$ are lost and only the message that carries the key information about packet P_6 arrives, the receiver node can still authenticate the keys of the other packets by deriving the key information supplied for P_6. Thus, though some of the packets may have been lost, the nodes can still authenticate them using the keys received. To do this, μTESLA has multiple phases that perform a particular job each, including Sender Setup, Broadcasting Authenticated Packets, Bootstrapping New Receivers, and Authenticating Broadcast Packets.

- *Sender setup*: In this phase, the sender wanting to broadcast messages in the WSN generates a one-way key chain, randomly selects the last key K_n, and generates the other values by applying the one-way function F on the chain for generating a length n. As F is a one-way function that any node can compute, the keys are generated forward but not backward; i.e., given K_{j+1} keys $(K_0, \ldots, K_j)$ can be computed, not K_{j+2}.
- *Broadcasting authenticated packets*: The sender uses the particular key for the corresponding time interval. For example, in the time interval I the sender uses the key K_1 and in the time interval $I + 1$, the node uses the key K_2. The packets sent in the particular time interval are authorized using the corresponding key.
- *Bootstrapping new receivers*: The keys in a one-way key chain are self-authenticating. If the receiver has one key in the key chain it can efficiently authenticate the other keys in the chain. If the receiver has value K_j in the key chain, it can easily authenticate K_{j+1}. Also, the sender and the receiver are required to be loosely time synchronized with the receiver having the knowledge of the sender's time disclosure schedule. Authenticating the key chain and having loose time synchronization establish strong freshness and point-to-point authentication. A receiver R sends a nonce N_R in the request message to the sender S, which replies to the message containing the following components: $T_S \rightarrow$ the current time of the sender, $K_i \rightarrow$ a key in the one way key chain, $T_I \rightarrow$ starting time, $T_{int} \rightarrow$ duration of time interval, and $\delta \rightarrow$ disclosure delay. The secret key shared between the node and the base station is used as the key for the MAC.
- *Authenticating broadcast packets*: An adversary sometimes knows the key used in the time interval I and may also have knowledge about the one-way key chain, so the receiver should ensure that the packet received is from an authenticated sender and not from an adversary before the key is released by that sender. This is achieved through loose synchronization of the sender and receiver. If the packet is legal, the receiver stores it; if it is spoofed, it is dropped. Once the receiver verifies the key, it authenticates the packets with the key and replaces that new key with the key it already has.

(3) Considerations for SPINS: SPINS uses less of a sensor node's memory; i.e., while crypto routines occupy 20 percent of the space, μTESLA occupies 574 bytes and 2 Kbytes is the acceptable total used memory [5]. The scheme's performance is also efficient, as the bandwidth

of the WSNs is adequate for the cryptographic primitives which SPINS uses. Additionally, most of the SPINS design may be used in other networks of low-end devices. Finally, the communication costs for SPINS are small, with security properties like data freshness, authentication and confidentiality only adding an overhead of 6 bytes in a 30-byte packet, which allows for inclusion in each packet [5]. SPINS can offer even greater advantages when restrictions on bandwidth and memory are slightly relieved.

Broadcasting and authenticating data are not very easy for individual nodes, as storing a one-way key in a node's memory is not possible, computation of the keys using a function generates much network overhead, and each node does not share a common key with every other node in the network. However, there are two solutions for this problem. First, the base station is used by a node to transmit all data that has to be broadcasted to other nodes. Second, the node broadcasts the data to the base station while the base station generates the authenticating keys using the one-way key chain. It is efficient to implement the cryptographic primitives in a single block cipher as WSNs are resource-constrained and, therefore, can not afford additional overhead for security. Yet, a strong cryptographic base is necessary for SPINS.

- *Block cipher*: Using RC5 can be very efficient in WSNs because of its small size and high efficiency. Moreover, as an algorithm it has been subject to scrutiny under many attacks. Using TEA could also work for block ciphers, but it is not subject to cryptanalysis scrutiny. DES and other algorithms are not usable for block ciphers due to their large size and high computation requirements that cannot be met in WSNs.
- *Encryption function*: The counter (CTR) mode of block ciphers can use the same function for both encryption and decryption, and the size of the cipher text is the same as the data in this mode. These two properties make this mode very useful while working in the encryption function of SPINS. Also, CTR mode offers semantic security, which is a strong cryptographic property already discussed. To use the CTR mode, both the sender and receiver nodes must maintain counters in their memory and possess an efficient way to synchronize the counters if needed. One advantage of maintaining a counter at both ends is that the messages now will not have an overhead of carrying the counter with them.
- *Freshness*: Using a counter and incrementing it every time a message is sent automatically provides weak freshness. For strong freshness, the sender must create a nonce and should include it in the request message to the receiver. SPINS uses a MAC function for generating random numbers and a counter is created to keep track of those created.
- *Message authentication*: Not only is a good encryption function necessary for data, but also a secure *MAC* is needed. As the block cipher is used more than once, *CBC-MAC* is used for *MAC*. An efficient way of message construction must be used to achieve authentication and message integrity. The construction $\{M\}_k$, $MAC(k',\{M\}_k)$, in which M is the data, K is the encryption key, and K' is the MAC key, is secure and protects the nodes from decrypting erroneous ciphered text.

Advantages of this scheme include that it is one memory efficient scheme, provides strong security features with less complexity, universal design allows use in many low-end devices, incurs low communication cost, and offers authentication and strong data freshness with a minimum overhead. Disadvantages of this scheme include μTESLA overhead from releasing keys after a certain delay, possible message delay.

3. Public key schemes

An MICA2 mote developed by the University of California at Berkeley has an 8-bit 7.3 MHz processor with 4 KB RAM and 128 KB of programmable ROM [26]. WSNs have mostly been using symmetric key and other non public-key encryption schemes [28]. A drawback to these schemes is that they are not as flexible as public-key schemes, but they are computationally faster. With limited memory, computing and communication capacity, and power supply, sensor nodes cannot employ sophisticated cryptographic technologies such as typical public key cryptographs. The use of public key cryptography on WSNs has not been tested enough to rule it out completely. Through the use of the MICA2 mote and TinyOS, public-key schemes are tested to determine their performance. Elliptic curve cryptography (ECC) has a faster computation time, smaller keys, and uses less memory and bandwidth than RSA [27]. Both ECC and RSA can be accelerated with dedicated co-processors. Recently, ECC has been used [25–28] for WSNs. The authors in [27] have implemented a way to execute public key schemes on WSNs with 8 MHz processors, using elliptic curves in computing encryption keys. The MICA2 mote is also able to use elliptic curve cryptography on their 8-bit processor. On these sensor nodes, public keys can be computed in less than 34 s using 1 K of RAM and 34 K of ROM [26].

3.1. Public-key schemes: RSA and ECC

Both RSA and ECC have been in research for many years. RSA stands for Rivest Shamir Adleman algorithm. It was developed in 1977 and is still one of the most popular public-key encryption technologies currently available. RSA relies on its strength due to the complication of factoring very large numbers. ECC was developed in 1985 independently by Koblitz and Miller. Its approach to public-key cryptography is based on the mathematics of elliptic curves. ECC can obtain the same security level as RSA while using a smaller key. A 160-bit ECC key has the same security as a 1024-bit RSA key [27]. A 224-bit ECC key compares to the 2048-bit RSA key [27]. This is due to the fact that it takes exponential algorithms to solve the elliptic curve discrete logarithm problem as opposed to small runtime algorithms to solve the large number factorization in RSA [27].

The RSA scheme generally introduces keys of size 512–2048 bits. Its takes a message M and composes a cipher text C using the key K. A method called the Chinese Remainder Theorem (CRT) can be used to accelerate RSA. Two prime numbers q and p are multiplied together to get the modulus n [27]. Computing these modular multiplications, CRT can lower computation time by almost 3/4 [27]. Other factors like the Montgomery multiplication and optimized squaring can reduce RSA complexity by 25%.

ECC is computed by point multiplication on elliptic curves over prime integer fields or binary polynomial fields [27]. The implementation of ECC on WSNs is primarily interested in prime integer fields since binary polynomial field mathematics is poorly supported by the slow processors. Operations of ECC scale linearly. This gives ECC an advantage over RSA on processors with small word sizes. Also, ECC grows in advantage as the key size grows.

CRT and modular multiplications for ECC and large integer mathematics for RSA are the most important operations in these cryptography schemes. Large numbers of multiplication operations need high memory read and writes due to the small word size of the processor. Computation time is therefore reduced by optimizing the number of memory operations [27].

ECC was implemented on two 8-bit platforms. Performance optimizations were applied due to limited resources. RSA-1024 and RSA-2048 was also implemented for comparison [27]. ECC-160 resulted with a private-key faster than RSA-1024. The performance was even more favorable when comparing ECC-224 to RSA-2048 [27]. ECC, on both platforms, outperforms RSA-1024 private-key operation. ECC also improves its performance over RSA as the word size of the processor decreases.

3.2. *TinyOS public-key implementation*

TinyOS comes as a part of the MICA2 mote. It has a link layer security mechanism based on SKIPJACK [26]. SKIPJACK is an 80-bit symmetric cipher introduced in 1994. The MICA2 offers access control, authentication, integrity, and confidentiality through TinySec, which is derived from SKIPJACK. Since TinySec allocates 80 bits for the key space, attackers could potentially have to do 2^{79} operations to find the key. TinySec actually uses 64-bit computed keys, with a 16-bit padding [26].

TinySec does not contribute much overhead to the MICA2 mote. On average, TinySec may decrease message transmissions by only 0.28 messages per second [26]. TinySec consumes about 7.9 KB of space on the mote. If programs do not need the entire 128 KB of programmable ROM and 4 KB of RAM, TinySec is a feasible security addition. Also, TinySec is not perfectly resilient to attacks. It relies on a single key, and therefore is unable to securely perform a rekey if necessary [26].

Discrete Logarithm Problem (DLP) is used to overcome TinySec's inability to securely distribute encryption keys. The Diffie–Hellman scheme is based off the DLP scheme. It allows two nodes to agree on a secret key over an insecure communication channel. Using a variant of the Diffie–Hellman scheme, two nodes can compute a key to use as the TinySec secret key [26]. The MICA2 motes version of Diffie–Hellman should include the modulus p of 1024 bits for security purposes.

The drawback of implementing this scheme is the computational overhead involved. Running the Diffie–Hellman scheme on a MICA2 mote with a modulus p being a 768-bit prime number takes 31 s [26]. If the modulus p is increased to 1024 bits, the computation time is 54.9 s. Computation of this magnitude is unacceptable in terms of energy requirements. The MICA2 mote, at full duty cycle, will decrease its lifespan to only a few days. Also note, the MICA2 mote uses 2 AA batteries, and dies when the voltage level drops below 2V [26].

Computations of this kind also take large amounts of memory. The public-key space is as large as the modulus p. To have pairwise keys for every node would exceed the memory space for the mote.

A MICA2 mote using ECC can effectively and securely distribute the 80-bit TinySec keys. ECC-163 is all that is needed [26]. ECC is as secure as Diffie–Hellman while using vastly smaller key sizes. ECC also offers perfect forward security.

The first attempt of implementing ECC on the MICA2 mote was based on the work of Michael Rosing, EccM 1.0 [26]. Initially, it took around 1.8 s to generate 33-bit keys. Unfortunately, larger key sizes caused the mote to reset due to stack overflows [26].

The second attempt utilized jBorZoi 0.9, to reach the 163-bit keys needed to compare with the Diffie–Hellman scheme. This contributes to EccM 2.0, which has a better success rate than the 1.0 version [26]. Other than successful compilation of 163-bit keys, EccM 2.0 uses less memory than EccM 1.0 and beats the runtime of the Diffie–Hellman scheme. Using EccM 2.0, it took on average 34.1 s to compute the 163-bit encryption keys [26].

3.3. *Key management using ECC*

With ECC, key management becomes easy. In [25], Du et al. propose a routing-driven key management scheme using ECC, which only establishes shared keys for neighbor sensors that communicate with each other. Both centralized and distributed key management schemes are proposed in [25].

4. Key predistribution schemes

In a key predistribution scheme, some keys are preloaded into each sensor before sensor deployment. After deployment, sensor nodes undergo a discovery process to set up shared keys for secure communications. This scheme ensures to some probability that any two sensor nodes can communicate using a pairwise key. This scheme does not, however, ensure that two nodes always are able to compute a pairwise key to use for secure communication.

In this section, we introduce many key predistribution schemes in the following subsections.

4.1. Random key predistribution scheme (Basic scheme)

In this subsection, we introduce Random Key Predistribution Scheme proposed by Eschenauer and Gligor [1]. This scheme is also referred to as Basic Scheme. First we will describe the structure and features of the Basic Scheme and then how it may be evaluated using two of the three Rs of efficient sensor networks, revocation and resilience (through rekeying). We will then analyze the pros and cons of the Basic Scheme for key establishment in a sensor network. In the Basic Scheme, key distribution is divided into three stages: key predistribution, shared-key discovery, and path-key establishment.

(1) key predistribution stage: In the *key predistribution stage*, a large key pool of |S| keys and their identifiers are generated. From this key pool, K keys are randomly drawn and pre-distributed into each node's key ring, including the identifiers of all those keys. At the point that each node has K keys and the identifiers of those keys, trusted nodes in the network are selected as controller nodes, and all the key identifiers of a key ring and the associated sensor identifiers on controller nodes are saved. Following this, the i-th contoller node is loaded for each node with the key that is shared with that node. This key predistribution process ensures that, though the size of the network is large, only a few keys need to be stored in each node's memory, thereby saving storage space. These few keys are enough to ensure that two nodes share a common key, based on a selected probability.

(2) Shared-key discovery stage: Once the nodes are initialized with keys, they are deployed in the respective places where they are needed, such as hospitals, war fields, etc. After deployment, each node tries to discover its neighbors with which it shares common keys. There are many ways for finding out whether two nodes share common keys or not. The simplest way is to make the nodes broadcast their identifier lists to other nodes. If a node finds out that it shares a common key with a particular node, it can use this key for secure communication. This approach does not give the adversary any new attack opportunities and only leaves room for launching a traffic analysis attack in the absence of key identifiers. More secure alternate methods exist for finding out the common keys shared between two nodes though. For example, for every key on a key ring, each node could broadcast a list $\{\alpha, E_{K_i}(\alpha), i = 1, \ldots, k\}$, where α is a challenge. The decryption of $E_{K_i}(\alpha)$ with the proper key by a recipient would reveal the challenge α and establish a shared key with the broadcasting node [1].

(3) Path key establishment stage: A link exists between two nodes only if they share a key, but the *path key establishment stage* facilitates provision of the link between two nodes when they do not share a common key. Let us suppose that node u wants to communicate with node v, but they do not share a common key between them. Node u can send a message to node y saying that it wants to communicate with node v; this message is then encrypted using the common key shared between node u and node y and, if node y has a key in common with node v, it can generate a pairwise key K_{uv} for nodes u and v, thereby acting like a key distribution center or a mediator between the communication of nodes u and v. As all the communications are encrypted using their respective shared keys, there will not be a security breach in this process. After the *shared-key discovery stage* is finished there will be a number of keys left in each sensor's key ring that are unused and can be put to work by each sensor node for path key establishment.

(4) Key revocation: A compromised sensor node can cause a lot of damage to a network and therefore, revocation of a compromised node is very important in any key distribution scheme. In the Basic Scheme, node revocation is conducted by the controller node. When a node is revoked, all the keys in that particular node key ring have to be deleted from the network. Let us assume that the controller node has knowledge about a compromised node in the network and broadcasts a message to all the nodes in the network, the message includes a list of the key identifiers of the compromised node's key ring. To sign the list of key identifiers, the controller node uses a signature K_e and then encrypts its message with K_{c_i}, which is the key that the controller node shares with the nodes during the *key predistribution stage*. Once each node receives the message, it decrypts the message using the key they already share with the controller node. When the signature is verified, the nodes search their key rings for the list of identifiers provided in the message and, if there is any match, the corresponding keys are deleted from the key ring. After the matching keys are completely deleted from all the nodes, there may be links missing between different ones and they then have to reconfigure themselves starting from the shared key discovery stage so that new links can be formed between them. As only few keys are removed from the network, the revocation process only affects a part of it and does not incur much communication overhead.

The keys used in a sensor network must be rekeyed to lessen the chance that an adversary may access all of the network keys when a few nodes and their keys are captured. Rekeying effectively increases a network's resilience without incurring much communication and computation overhead.

(5) Analysis of the basic scheme: Let us assume that the probability of a common key existing between two nodes in the network is p, and the size of the network is n. The degree of a node d is derivable using both p and n since the degree of any node is simply the average number of

edges connecting that node with other nodes in its neighborhood; therefore, $d = p \times (n - 1)$. First we have to find the value of d such that a network of n nodes is connected with a given probability P_c. We then must calculate the key ring size k and the size of the key pool $|S|$.

According to Random Graph Theory, a random graph $G(n,p)$ is a graph consisting of n nodes and p representing the probability of establishing a link between two nodes. Erdos and Renyi [12] showed that there exists a probability state p, which moves from state zero to state one for large random graphs. The function that defines p is called the threshold function of a property. If we are given a desired probability (P_c) for graph connectivity, then p is given as $P_c = \lim_{n\to\infty} \Pr[G(n,p) \text{is connected}] = e^{e^{-c}}$, $p = \frac{\ln(n)}{n} + \frac{c}{n}$, where c is a real constant.

Then, to calculate the key ring size k and the size of the key pool $|S|$, we need to first note that wireless constraints limit the number of nodes in a range to be smaller than n, represented by the value n'. Now the probability of sharing a key between two neighbor nodes varies to $p' = d/(n' - 1)$, for a given d value. Also, p' can be denoted as the difference between the total probability and the probability that two nodes do not share a common key; i.e., $p' = 1 - Pr$[two nodes do not share any key] and, thus, $p' = 1 - \frac{(1-\frac{k}{p})^{2(p-k+\frac{1}{2})}}{(1-\frac{2k}{p})^{(p-2k+\frac{1}{2})}}$, where $|S|$ is the size of the key pool and k is the key ring size.

Eschenauer and Gligor [1] have shown that for a pool size $S = 10{,}000$ keys, only 75 keys need to be stored in a node's memory to have the probability that they share a key in their key rings to be $p = 0.5$. If the pool size is ten times larger, i.e., $S = 100{,}000$, then the number of keys required is still only *250*. Thus, the Basic Scheme is a key management technique that is scalable, flexible and can also be used for large networks. Trade-offs in the Basic Scheme can be made between sensor memory and connectivity, but it does not provide the node-to-node authentication property that ascertains the identity of a node with which another node is communicating. This property is very useful when revoking misbehaving nodes from the network and also helps in resisting the node replication attack.

Many key management schemes [2,3,8,31–33], etc.) are proposed as extensions of the Basic Scheme to make it even more secure and reliable.

Advantages of this scheme include flexible, efficient, and fairly simple to employ, while also offering good scalability. Disadvantages of this scheme include that it cannot be used in circumstances demanding heightened security and node-node authentication.

4.2. Q-composite random key predistribution scheme

Chan, Perrig, and Song [2] have introduced two variations of the Basic Scheme, Q-Composite Random Key Predistribution and Multipath Key Reinforcement, and a variation of the commonly known Pairwise Scheme, called Random Pairwise Scheme. Each comes with a different kind of trade-off and is not, therefore, widely applicable. The Q-Composite Scheme achieves security under small scale attacks while being vulnerable under large scale attacks and is useful where large scale attacks are easily detected. We will introduce Multipath Key Reinforcement scheme and Random Pairwise Scheme in the next two subsections.

In the Basic Scheme, two nodes share a unique key for establishing a secure communication link. A given network's resilience to node capture can be improved by increasing the number of common keys that are needed for link establishment. The Q-Composite Random Key Predistribution Scheme does this by requiring that two nodes have at least q common keys to set up a link [2]. As the amount of key overlap between two nodes is increased, it becomes harder for an adversary to break their communication link. At the same time, to maintain the probability that two nodes establish a link with q common keys, it is necessary to reduce the size of the key pool $|S|$, which poses a possible security breach in the network as the adversary now has to compromise only a few nodes to gain a large part of S. So the challenge of the Q-Composite Scheme is to choose an optimal value for q while ensuring that security is not sacrificed.

In the *key predistribution stage* of both the Basic and Q-Composite Schemes, k random keys are picked from S and initialized in each node's key ring. In the *shared-key discovery phase*, each node has to find the common keys which it shares with other nodes by either making all the nodes broadcast their key identifiers or by selecting a slower and more secure method of posing puzzles such as the Merkle Puzzle [18]. For this puzzle method, each node issues m client puzzles to each neighboring node and any node that comes up with the correct solution to the puzzle is identified as sharing the associated key. After this, the two schemes differ in that the Q-Composite Scheme requires each node to identify neighboring nodes with which they share at least q common keys while the Basic Scheme only requires one shared key. This restriction in the Q-Composite Scheme allows the number of keys shared to be more than q but not less, represented by the value q. At this stage in the process, nodes will fail to establish a link if the number of keys shared is less than q; otherwise, they will form a new communication link using the hash of all the q keys, i.e., $K = hash(k_1 \| k_2 \| \ldots \| k_q)$.

S, the size of the key pool, is the critical parameter that must be calculated for the Q-Composite Scheme to be efficient. If S is large, then the probability that two nodes share a common key and therefore can communicate is decreased. However, if S is decreased, an adversary's job may be easier as she can gather most of the keys in the key pool by capturing only a few nodes. Thus, S must be chosen such that the probability of any two nodes sharing at least q keys is is larger than or equal to p.

Chan, Perrig, and Song's [2] method to calculate S is $p(i) = \frac{\binom{|S|}{i}\binom{|S|-i}{2(m-i)}\binom{2(m-i)}{m-i}}{\binom{|S|}{m}^2}$, where $p(i)$ is the probability that any two nodes have exactly i number of keys in common; and m is the key ring capacity for a given node. There are $\binom{|S|}{i}$ ways to pick i and $|S| - i$ is the number of the remaining keys in the key pool after i is picked. There are $\binom{|S|}{m}$ different ways to pick m and $\binom{|S|}{m}^2$ total number of ways for both nodes to pick m.

Also, to assign the remaining keys $2(m - i)$ distinct keys are picked from the key pool for each node and the number of ways to do this is $\binom{|s| - i}{2(m - i)}$. There are $2(m - i)$ ways to partition the keys equally between the two nodes.

Let P_c be the probability of any two nodes sharing sufficient keys to form a secure connection. Therefore, $P_c = 1 -$ (the probability that the two nodes share insufficient keys to form a connection) or $P_c = 1 - (p(0) + p(1) + \ldots + p(q - 1))$. Now the largest $|S|$ such that $P_c \geqslant p$ is chosen.

The evaluation of the Q-Composite Scheme can be done by verifying its resilience to node capture. Even though this scheme does not provide resistance to node replication or a means for node-to-node authentication since the keys from the key pool are used more than once, it does improve resilience to node capture when an adversary has successfully captured some other nodes in the network. As the same keys are used repeatedly in a network, a situation may arise in which two nodes effectively have their communications exposed due to the compromise of other two nodes that share the same key(s). Chan, Perrig, and Song [2] have calculated the probability that a secure link that is established between two uncompromised nodes will be compromised as $\sum_{i-q}^{m}(1 - (1 - \frac{m}{|S|})^x)^i \frac{p(i)}{p}$, where x is the number of nodes captured, i is the number of keys in common, m is the number of keys in the key ring of a node, and p is the probability of setting up a secure link.

The Q-Composite Scheme offers greater resilience compared to the Basic Scheme when a small number of nodes have been captured in the network. The amount of communications that are compromised in a given network with the Q-Composite Scheme applied is 4.74 percent when there are 50 compromised nodes, while the same network with the Basic Scheme applied will have 9.52 percent of communications compromised. Though the Q-Composite Scheme performs badly when more nodes are captured in a network, this may prove a reasonable concession as adversaries are more likely to commit a small-scale attack and preventing smaller attacks can push an adversary to launch a large-scale attack, which is far easier to detect.

Still, random key predistribution schemes like Q-Composite and Basic cannot be securely used for large networks because they use keys more than once, which results in the compromise of a larger fraction of communications when just a few nodes are compromised. Since random key predistribution schemes are not scalable, the maximum network that can be supported should be measured using the *Limited Global Payoff Requirement*, which states that given a secure network, an adversary should not learn anything about the communications of nodes in the network other than those of captured nodes. Let f_m be the maximum compromise threshold above which the adversary gains an unacceptable high confidence of guessing the sensor readings of the entire network. If x_m is the number of nodes compromised, then the total fraction of secure links compromised after the key setup phase due to these x_m nodes being compromised is $f(x_m)$; if the total fraction of secure links compromised reaches the threshold value with the x_m nodes being compromised, i.e., $f_m = f(x_m)$, then Chan, Perrig, and Song [2] have calculated the maximum allowable size of the network to be $n \leqslant 2x_m(1 + \frac{1}{f_m})$. For example, when $p = 0.33$, $f_m = 0.1$, and $m = 200$, the maximum supportable network size for a Q-Composite Scheme ($q = 2$) is 1,415 nodes. Compared to the 1,159 node maximum of the Basic Scheme, the advantage is obvious. Thus, the Q-Composite Scheme is more efficient than the Basic Scheme in providing more resilience to node capture and significantly increases the maximum allowable size of a network. Since both schemes fail to provide node-to-node authentication or resistance against node replication, it is important to review other schemes that work more efficiently in networks requiring such security measures.

Advantages of this scheme include that it provides better security than the Basic Scheme by requiring more keys for two nodes to share one for communication, which makes it difficult for an adversary to compromise a node. Disadvantages of this scheme include that it is vulnerable to breakdown under large-scale attacks, and does not satisfy scalability requirements.

4.3. Multipath key reinforcement scheme

The Multipath Reinforcement Scheme [2] offers good security with additional communication overhead for use where security is more of a concern than bandwidth or power drain.

The idea of using a multipath to reinforce links in a random key establishment scheme was first explored by Anderson and Perrig [10]. Chan, Perrig, and Song [2] further developed the Multipath Key Reinforcement Scheme for establishing a link between two nodes of a given network that is stronger than that in the Basic Scheme. The links formed between nodes after the key discovery phase in the Basic Scheme are not totally secure due to the random selection of keys from the key pool allowing nodes in a network to share some of the same keys and, thereby, possibly threaten multiple nodes when only one is compromised. To solve this problem, the communication key between nodes must be updated when one is compromised once a secure link is formed. This should not be done via the already established link, as an adversary might decrypt the communication to obtain the new key, but should be coordinated using multiple independent paths for greater security.

If node A needs an updated communication key with node B, all possible disjointed paths to node B must be used. Assume that there are h such disjointed paths from node A to node B. Then node A generates h random values $(g_1, g_2, \ldots, g_h)$, each equal to the size of an encryption key, and sends one down each available disjointed path to node B. When node B has received all h random values, it computes the new encryption key at the same time as node A does form a new and secure communication link with $k' = k \oplus g_1 \oplus g_2 \oplus \ldots \oplus g_h$, where k is the original key.

With the new link in place, the only way an adversary can decrypt the communications is to compromise all the nodes involved in the formation of the key. The larger h is, the more paths and nodes involved and the greater the security of the new link. This increase in network communications causes excessive overhead in finding multiple disjointed paths between two nodes. Also, as the size of a path increases, it may grow so long as it leaves a chance for an adversary to eavesdrop, which makes the whole path insecure. A *2-hop approach* to the Multipath Key Reinforcement Scheme considers only 2-link paths to minimize the overhead of path length by using disjointed paths that are only one intermediate node away from the two original nodes (A and B).

To take such an approach, first the number of common neighbors between the original nodes must be calculated in a planar deployment of sensors. In [2], Chan, Perrig, and Song calculated the overlap of communication radius between two nodes in a network to be $0.5865\pi\, r^2$ in a planar deployment where r is the communication range of sensors; therefore, the expected number of common neighbors with whom both nodes share a secure link is $0.5865p^2n'$, where p is the probability of sharing sufficient keys to communicate and n' is the number of neighbors in each node. Expressed as $0.5865d^2/n = k$, both nodes share a secure link with an expected k neighbors, e.g., if $d = 20$ and $n' = 60$, then $k = 3.91$. To assess the efficiency of the *2-hop approach*, the new probability for compromising the link between two nodes needs to be derived. If an adversary's basic probability of compromising the 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 the probability of compromising both hops in the path or $2b - b^2$ [2]; hence, the final probability of breaking the link will be $b' = b(2b - b^2)^k$.

If $b = 0.1$ and the number of neighbors (k) is 3, then the chance of eavesdropping after reinforcement improves to 6.86×10^{-4}, that is about 1 in 1,458. Chan, Perrig, and Song [2] calculated the total additional communication overhead incurred to be at least $2 \times 0.5865p^2n'$ times more in the 2-hop approach compared to the normal setup. If, for instance, $p = 0.33$ and $n' = 60$, additional overhead can be at least 7.66 times. Given these results, the minimal network overhead of finding the neighbors that share a common key becomes a reasonable trade-off when using a *2-hop* Multipath Key Reinforcement Scheme to increase the security of wireless sensor networks, although this scheme remains constrained by certain vital factors, including the deployment density characteristics of the network.

Advantages of this scheme include that it offers better security than the Basic Scheme or the Q-Composite. Disadvantages of this scheme include that it creates communication overhead that can lead to depleted node battery life and to the chance for an adversary to launch DOS attacks.

4.4. Random pairwise key scheme

Compared to the Q-Composite scheme and the Multipath scheme, the Random Pairwise Scheme [2] offers the best security features in its resilience to node capture with the only drawback being limited scalability.

Node-to-node authentication not only helps to reduce overhead since sensor nodes instead of a base station take actions when a node is compromised in the network, but also entails that each node use a unique identity, which helps nodes to identify exactly which ones are compromised. We have seen that though the Basic Scheme is somewhat efficient, it does not provide node-to-node authentication. The Q-Composite extension and 2-hop Multipath approach do not provide node-to-node authentication either. As we introduced before, the Pairwise Key Establishment Scheme, however, is one of the most efficient key establishment schemes because it does offer many additional features compared to other schemes, including node-to-node authentication and resilience to node replication. However, since sensor nodes are resource-constrained, this significant overhead limits the scheme's applicability, but it can be effectively used for smaller networks.

Chan, Perrig, and Song [2] developed the Random Pairwise Scheme as an extension of the Pairwise Scheme to help overcome this drawback. They stated that not all $n - 1$ keys are required to be stored in a node's key ring. As we have already seen with the Basic Scheme, not all nodes must be connected as long as node connections meet some desired probability P_c, which dictates that only np keys are needed to be stored in a given node's key ring, where n being the number of nodes in the network and p being the probability that two nodes can communicate securely. Given this, if k is the number of keys in a node's key ring, the maximum allowable network size can be determined with $n = k/p$ for the Random Pairwise Scheme.

In the initialization stage of this scheme, n unique node identifiers are created, each paired with m other randomly selected node identities. A pairwise key is then generated for each such pair. Both the generated pairwise key and the identity of the other node that shares the key are stored in each node's memory. Additional identifiers can be generated to allow for scalability of the network, i.e., the number of nodes may originally be fewer than n created to allow for adding nodes. In the key discovery phase, each node broadcasts its identity to the other nodes in the network. For example, if node A wants to communicate with other nodes in the network, it broadcasts its identity to other nodes in the network; if the neighboring nodes share a pairwise key with node A, they perform a cryptographic handshake with node A, thereby forming a secure communication link. This process of broadcasting can also be extended beyond the communication range of a node by making the intermediate nodes rebroadcast the node identity to a certain number of hops, which in turn helps in increasing the maximum allowable size of the network. This process of range extension must be done cautiously as it leaves a vulnerable opening for an adversary to perform a denial of service attack. The attack involves the adversary's introducing foreign nodes into the network to generate random node identities that flood the network with rebroadcasted identities, making the whole scheme slow and inefficient. This type of attack can be avoided by restricting the number of hops for range extension.

Revoking compromised nodes from a network helps avert various attacks such as denial of service, implanting clones, dropping legitimate reports, etc. Revocation of sensor nodes through the base station can be a slow process due to the high latency in communications with the sensor nodes. To overcome this difficulty, Chan, Perrig, and Song [2] also developed a distributed node revocation method for the Random Pairwise Scheme. Assume that the scheme can detect compromised nodes. If node A finds a certain node B to be compromised then it casts a public vote against node B. If a threshold of t such votes have been cast against node B by other nodes in the network, node A will disconnect all its communication with node B. This process continues until all the nodes in the network break their links with node B, thereby 'deleting' node B from the network. All the nodes that vote against node B are called the 'voting members' of node B and, as node B shares exactly k pairwise keys with other nodes, there will be k voting members of node B.

This voting method of node revocation must have certain important properties to function properly: make the broadcasted public votes without replay value, disallow a voting member from forging another vote, provide a means for each voting member to verify the validity of the votes that are being broadcasted, etc. In this voting method, each of the k voting members of node B is initialized with a random key K_i, and should know the hash values of the remaining $k - 1$ voting members. To revoke node B from the network, node A broadcasts its K_i key. All other voting nodes verify the key by calculating the hash value of the key. Once verified, the key is replaced with a flag signifying the vote has already been used.

Given this process, the nodes in the network must store an additional $k - 1$ hash values, a voting key and the pairwise keys, which drastically increases the overhead in a sensor node's memory. Chan, Perrig, and Song [2] proposed using a Merkle Tree [11] to authenticate the k hash values to reduce overhead by requiring verification and storing of only one hash value, which reduces the memory size required on the node, but also increases the size of the voting information to $O(log(k))$ as each node must still recall which ones have already been received from public vote to remove possible replay.

Other precautionary measures with the Random Pairwise Scheme taking the public voting approach include the critical issue of choosing the threshold or t value. If t is high, there may not be enough neighboring nodes to revoke a node that has been compromised; however, if t is low, then a group of compromised nodes may cause the revocation of many legitimate nodes. For instance, a network of 1,000 to 10,000 nodes should have a t value from 1 to 5. Also, since this approach requires a node have at least t neighbors in its communication range to be revoked, an adversary can attack the network by selectively disrupting a given node such that only $t - 1$ legitimate nodes are able to communicate with it so that it cannot be revoked.

Beyond problems with t value, in the public voting approach every vote that is cast is being transmitted to all the nodes in the network, which may lead to a denial of service attack. To solve this, only the voting members should be required to rebroadcast votes between each other while the remaining nodes are forced to ignore the communication. In this way, the degree of vulnerability to falsely rebroadcasted identities can be decreased. Additionally, the node that first receives the correctly verified vote rebroadcasts it only a fixed number times to increase the probability of successful transmission to neighboring voting members.

In a similar action, an adversary that tries to compromise a fixed number of nodes can compromise a significant portion of a network when public voting is used to perform distributed node revocation since each node can potentially cast a vote against k others. To prevent this problem, only nodes that establish direct communication are given the ability to revoke a compromised node by distributing masked revocation keys to voting members in a non-working form. Each node would then complete the key discovery phase by sharing the secret key only with other nodes with whom they already share a pairwise key connection.

Advantages of this scheme include that is offers the best security of all the above schemes with perfect resilience to node capture as the keys used by each node are unique, and also provides resistance against node replication. Disadvantages of this scheme include that it does not support networks of large size, and does not satisfy scalability requirements.

4.5. Polynomial pool-based key predistribution

Every key distribution scheme previously discussed has one or more trade-offs to be considered and what they fundamentally lack is a greater probability of key establishment despite part of the network being compromised. To this end, Liu and Ning [3] proposed the Polynomial Pool-Based Key Predistribution Scheme that offers several efficient features the other schemes lack, including:

- Any two sensors can definitely establish a pairwise key when there are no compromised sensors;
- Even with some nodes compromised, the others in the network can still establish pairwise keys;
- A node can find the common keys to determine whether or not it can establish a pairwise key and thereby help reduce communication overhead.

In the initialization stage of the Polynomial Pool-Based Scheme, the setup server randomly generates a bivariate t-degree polynomial $f(x,y)$ over a finite field F_q, where $f(x,y) = \sum_{i,j=0}^{t} a_{ij}x^i y^j$.

The value of q is a prime number which can accommodate a cryptographic key. The equation $f(x,y)$ has the property $f(x,y) = f(y,x)$. The setup server then generates a polynomial share of the equation for every node in the sensor network; e.g., node i in the network receives an $f(i,y)$ share and node j receives an $f(j,y)$ share. If both nodes i and j want to establish a common key $f(i,j)$ between them, then node i can compute the common key by computing $f(i,y)$ at node j and then node j can compute $f(j,y)$ at node i for the common key $f(i,j)$. This methodology is secure and reveals nothing about the communication between other nodes until t nodes have been compromised, making it t collusion resistant where the t value depends upon the memory available in the sensors.

Each node in this scheme must store a t-degree polynomial which occupies $(t+1)log(q)$ storage space. Increasing the size of the network increases the chance of compromising more than t nodes, but modifications based on the Basic Scheme can earn good results. For this, instead of using a single t-degree polynomial, a pool of polynomials is used. During the initialization phase, randomly selected polynomials are deployed into each node's memory. When there is only one polynomial remaining in the pool, the scheme falls back to the Polynomial Pool-Based Key Distribution; but, if all of the polynomials are 0-degree, then distribution resembles the Basic Scheme.

In the *key predistribution stage* of the Polynomial Scheme, the setup server generates a set of bivariate t-degree polynomials over a field F_q. Each polynomial is then assigned with a particular ID for the server. A subset of these polynomial shares are then picked up by the server and placed in each of the network's nodes. While polynomial placement is the main issue of this stage, in the *key discovery stage* each sensor node finds a node with which it shares the same bivariate polynomial and both nodes establish a common key. The complex issue is to find whether two nodes share the same polynomial or not, for which there are two techniques: *predistribution* and *real-time discovery*.

In the predistribution approach, the knowledge of the nodes with which each node will share a polynomial is pre-loaded. This is a basic method in which each node carries the node IDs of those with which they share a polynomial. The concessions of this method are that it does not offer the flexibility of adding new nodes into a network and it leaves the network vulnerable to attack. Since information is predistributed in this approach, an adversary may attack a node and gain access to the stored data, which would help in targeting certain nodes in the network. Conversely, nodes must uncover with which others they share a polynomial after deployment when applying the real-time discovery method. This discovery can be done by broadcasting the IDs of the polynomials that nodes share or by challenging the nodes with puzzles that are only solvable if the nodes share part of the bivariate t-degree polynomial. Even though this handles problems faced with the predistribution approach, real-time discovery increases the communication overhead of the network, which makes weighing these factors critical in choosing a method.

After key discovery, if two nodes do not find a common polynomial share, they must communicate through a path key. If node P wants to communicate with node Q and the two nodes do not have a common polynomial share, node P must find a path through which it can communicate with node Q and either node can then send a request to establish a pairwise key for communication. The problem with this stage is that intermediate nodes should be able to communicate with both nodes and, similar to the previous stage, there are two customary techniques for finding intermediate nodes: *predistribution* and *real-time discovery*.

In the predistribution approach, the setup server preloads each node with information such that, if a node is given an ID, each node can find a path to it. In this stage, the predistribution method suffers from the same problems as faced in key predistribution no scalability and vulnerability to attack. With real-time discovery, nodes try to find a communication path on-the-fly. A source node sends a message to adjoin intermediate nodes that it wants to establish a pairwise key with the destination node and, as the source node already shares a common key with the intermediate nodes, there is no security threat in this communication. If an intermediate node of the source node shares a common key with the destination node, then a communication path has been discovered between the source and destination nodes through which they may discover a common key. Again, the concession of real-time discovery is additional overhead of communication.

Polynomial Pool-Based Key Predistribution using random subsets offers greater security and flexibility when compared to other schemes until a certain number (60 percent) of compromised nodes has been reached at which point any scheme would prove ineffective. Compared to the Random Pairwise Scheme, which offers perfect resilience to node capture as no key in the network is used twice, the Polynomial Pool-Based offers the same resilience if a polynomial share is used no more than t times. Also, the Polynomial Pool-Based Scheme offers certain advantages over Random Pairwise in that sensors can be added dynamically without consulting the already deployed sensors while dynamically deploying nodes in Random Pairwise demands that the server has predesignated unassigned space for additional nodes, which may never be deployed. Because of this, the Random Pairwise Scheme can only offer limited scalability, while the more attractive Polynomial Pool-Based Scheme allows for undetermined network growth.

Advantages of this scheme include that it allows the network to grow to a larger size after deployment. Disadvantages of this scheme include t-collision resistance (compromising more than t polynomials leads to network compromise).

4.6. Random subset key predistribution

Random Subset Scheme [3] is an extension of the Polynomial Pool-Based Scheme using a random subset key assignment and the Basic Scheme [1], in which random keys are selected from a large key pool and then assigned to each node in a network. In the Random Subset Scheme, random polynomials are selected from a polynomial pool and assigned to each node in a network to avoid the Basic Scheme's vulnerability in possibly using a key in more than one node. With the Random Subset scheme, the pairwise keys generated by each node are unique and based upon the each node's ID. If no more than t shares of the same polynomial have been disclosed, it is very difficult to attack the communication between two nodes.

The Random Subset scheme works similarly to the Polynomial Pool-Based scheme in the three stages of key establishment. In the key predistribution stage, the setup server generates a set F of s-bivariate t-degree polynomials and then initializes each node with a subset of s' polynomials from F. In the key discovery stage, each node attempts to determine the nodes with which they share a common key by employing the real-time discovery technique as information is not preloaded in the nodes prior to deployment. In the path key establishment phase, a source node sends a message to its intermediate nodes seeking to establish a connection with a destination node and, if an intermediate node shares a common key with both the source and destination nodes, then a communication path is formed between the two. Generally, the communication range of the source node is limited to lessen vulnerability.

The probability of two sensors sharing the same bivariate polynomial is the same as the probability of the two sharing a common key as described in the Basic Scheme discussion, $p = 1 - \prod_{i=0}^{s'-1} \frac{s-s'-i}{s-i}$.

This can also be applied to calculate the probability that any two sensors can establish a common pairwise key using both the key discovery and path discovery stages. If there are d neighbors to a node and any one of them can act as the intermediate node, the probability that one of them share a common key with the source and destination will be p^2; therefore, the probability that the sensor nodes establish a pairwise key in either the key discovery or the path key establishment stage will be $P_s = 1 - (1-p)(1-p^2)^d$.

If $p = 0.3$ and $d = 30$, then $P_s = 0.959$. Assuming that an attacker has compromised N_c sensors in the network, where $N_c > t$, the scheme is known to be secure until the adversary compromises fewer than t sensors. With a pool of F polynomials, the probability that a polynomial is used i times and its probability of being compromised when more than t nodes are compromised must be calculated to determine security efficiency. Given this, the probability that a polynomial is chosen for a sensor node will be s'/s and the probability that this polynomial is chosen exactly i times among the N_c compromised nodes is $p(i) = \frac{N_c!}{(N_c-i)|i|}(\frac{s'}{s})^i(1-\frac{s^i}{s})^{N_c-i}$. Thus, the probability of a polynomial being compromised is $P_c = 1 - \sum_{i=0}^{t} p(i)$.

Even though this makes the network more secure, it is still vulnerable to attack because if an adversary somehow knows the distribution of polynomial shares, specific nodes can be targeted for attack to compromise communications. It is enough for an adversary to compromise $t+1$ particular nodes to compromise a polynomial and, in effect, the network. This problem is addressed by restricting the use of polynomial shares to a maximum of $t+1$ times in the network so that an attacker must now compromise all the $t+1$ nodes to compromise a polynomial. Though efficient, use of random subsets like this decreases the potential size of a network. The maximum number of nodes in a network when random subsets are implemented is $(t+1)s/s'$; however, using this scheme is unnecessary as it is relatively difficult for an adversary to compromise $t+1$ selected nodes.

4.7. Grid-based key predistribution

A Polynomial Pool-Based Scheme [3] using a grid-based key assignment offers all the attractive properties of the Polynomial Pool-Based key predistribution and guarantees that two sensors can establish a pairwise key when there are no compromised nodes and the nodes can communicate with each other. Even if some nodes are captured, there will still be a great chance for key establishment between uncompromised nodes using this approach, which also reduces network communication overhead. With grid-based key predistribution, a sensor node can determine whether it can establish a pairwise key with another node or not, and can say which polynomial should be used for key establishment.

If a network consists of N sensor nodes, the approach involves constructing an $m \times m$ grid with a set of $2m$ polynomials, calculated as $\{f_i^c(x,y), f_i^r(x,y)\}, i = 0, \ldots, m-1$, where the value of m is the square root of N; each row i in the grid is associated with a polynomial $f_i^r(x,y)$ and each column of the grid is associated with a polynomial share $f_i^c(x,y)$. The setup server distributes an intersection in the grid to each node, and then distributes the polynomial shares of that particular column and row to the node to provide each node with the information required for key discovery and path key establishment. Although the Grid-Based Scheme can be extended to n-dimension, Liu and Ning [3] considered only a 2-dimension with many polynomials, so that is the example discussed here.

In the first stage of key establishment, the setup server generates $2m$ t-degree bivariate polynomials over a finite field F_q and assigns each node to an unoccupied intersection in the grid for deployment in the network. If the intersection is $\langle i,j \rangle$, then the node ID is $\langle i,j \rangle$. The server provides each node with its ID and the row and column polynomial shares of that grid intersection. To facilitate path discovery, all nodes are densely placed in a rectangular area in the grid.

In the second stage, polynomial share discovery, if node i wants to establish a pairwise key with node j, it checks for common rows or columns with j, i.e., $c_i = c_j$ or $r_i = r_j$. The pairwise key can be established using the polynomial shares of a row or column that matches. Should none match, then nodes i and j must find an alternate path to each other in the path key establishment stage. To do so, node i finds an intermediate node through which it can establish a pairwise key with node j. Even if some intermediate nodes are compromised, node i can still find a path to node j since there are many connecting paths in the grid between the two nodes; but, as the number of compromised nodes increases, so does the length of the path.

In this case the nodes remember the graph composing the grid; however, with large networks, it is not feasible for a node to remember the entire graph or run an algorithm for finding the path between the nodes. Discovering key paths using two intermediate nodes limits the demands on the nodes for this scheme to function in large networks also. If, for instance, there are two sensor nodes attempting to establish a path key between them, the source node S determines the set of N nodes with which it can communicate, and then selects some nodes randomly from that set. Node S also generates a random number r and a counter c. Node S sends each node U of the subset N a message containing the IDs of nodes S and D, the counter value c and K_c in an encrypted form.

Here the value of $K_c = F(r,c)$ where F is a pseudo random function. Encryption of the message is done using the pairwise key that node S shares with the intermediate node U. After receiving the message from node S, node U checks for the authentication of the message and if the message is authenticated, node U attempts to find a non-compromised node V. It then sends to node V the message sent by node S in the encrypted form using the pairwise key which it shares with node V. If node V receives the message and discovers that it can establish a pairwise key with node D, it sends the message to node D in encrypted form using the shared key. Once the destination node D receives the message, it knows that node S wants to establish a pairwise key with it and then sends node S the counter value c and the new communication key $K_{s,d} = K_c$.

With grid-based key predistribution, an adversary may try to attack the connection between two nodes by either compromising the pairwise key or by preventing the two nodes from establishing a shared key. If an adversary wishes to attack the entire network, the foe may attempt to lower the probability of establishing a pairwise key between nodes. This may be done through attacking a pair of nodes and finding their common key without actually compromising the nodes by compromising the polynomial which the two nodes share. To discover the polynomial share, the adversary must compromise at least $t + 1$ nodes as stated previously. The network may avert such an attack even when the adversary successfully finds out the polynomial which two nodes share by the nodes' establishing a common key through path key establishment.

From the scheme we can see that there are still $m - 1$ nodes that can help nodes U and V establish a common key. An adversary must compromise at least one node in each pair to arrest path key establishment; thus, the adversary must compromise $t + 1$ nodes to learn the pairwise key and $t + m$ sensor nodes to prevent two from establishing a pairwise key via intermediates. An adversary can also compromise the polynomial shares in a pool by knowing the subset assignment mechanism. Supposing that the adversary has compromised some l polynomials from the pool, there are about ml sensors with at least one polynomial share disclosed. The attacker has compromised about $(t + 1)l$ sensor nodes, but only affects the common keys in ml sensors, including those of the compromised nodes. The adversary may also attack the sensors randomly to disrupt path establishment and thereby make key establishment an expensive process. If P_c nodes have been compromised, then the probability that exactly k polynomial shares on a particular polynomial are disclosed is $P(k) = \frac{m!}{k!(m-k)!} p_c^k (1 - p_c)^{m-k}$. The probability that one particular polynomial is compromised would be calculated as $P_c = 1 - \sum_{i=0}^{t} p(i)$.

This grid-based approach to the Polynomial Pool-Based Scheme has reasonable overhead when compared to other schemes. Each node must store 2 bivariate t-degree polynomials and IDs of the compromised nodes with which it can establish a pairwise key; therefore, the total overhead for each node is, at most, $2(t + 1)log(q) + 2(t + 1)l$ bits. The network overhead is almost null when there is direct key establishment between nodes. There is slight communication overhead when two nodes must find a common key through path key establishment and this overhead increases with each additional node compromised. The approach offers many attractive properties other schemes do not, including nice resilience to node capture until a certain percentage of nodes are compromised (60 percent). The Basic Scheme and Q-Composite Scheme offer this same resilience, but the grid-based approach offers less overhead on both network communication and computations. Compared to Random Pairwise Scheme, the grid-based method offers the same degree of security when the same number of sensors and storage overhead are considered. More than any other scheme, the grid-based approach offers greater probability of key establishment when there are no compromised nodes as well as greater probability of key establishment with some nodes compromised.

Finally, there will be a greater chance for nodes to establish a pairwise key with others without communication overhead as the sensors are deployed in a grid-like structure.

Thus, the Polynomial-Based Key Predistribution and the two instantiations of the scheme, polynomial pool-based and grid-based, provide some attractive and efficient properties for key establishment in networks. In the near future, these schemes might be further extended. For example, the grid-based approach may be extended into *n*-dimension or hypercube-based. Also, research must be done for networks with Polynomial-Based Key Predistribution scheme and mobile properties.

Advantages of this scheme include that it offers low communication and computation overhead on the network, guarantees a total connected graph when there are no compromised nodes and all the nodes are in transmission range. Disadvantages of this scheme include overhead of node storage as each must not only share the polynomial keys but also the ID's of compromised nodes to avoid attack.

4.8. Hypercube key distribution scheme

Hypercube Key Distribution Scheme [31] guarantees that any two nodes in the network can establish a pairwise key if there are no compromised nodes present as long as the two nodes can communicate [31]. Also, nodes can still communicate with high probability if compromised nodes are present. Nodes can decide whether or not they can directly communicate with other nodes and what polynomial they should use when transmitting messages [31].

If we denote the total the number of nodes in the network to *N*, then this scheme computes an *n*-dimensional hypercube with m^{n-1} polynomials [31]. Before node distribution, a setup server assigns each node an exclusive coordinate in a matrix. Also the setup server assigns each node a set of polynomials in which it can compute a pairwise key with other nodes for communication.

To compute the initial polynomials, the setup server makes $n \times m^{n-1}$ number of polynomials over a space of F_q. Each node occupies an empty space in the matrix. If nodes *a* and *b* share a common polynomial, they can make a direct connection and compute a pairwise key to communicate. If the two nodes do not share a common polynomial, they have to use the path discovery method to compute an indirect key [31].

(1) Dynamic path discovery: Predetermined paths can be used for pairwise key generation for nodes that are unable to communicate directly. Also it is feasible for a node to flood the network to find a key path but this is impractical due to the resource constraints on the sensor nodes [31].

An alternative path discovery algorithm described in [31] finds paths between nodes *a* and *b* dynamically. In this method, the source and other nodes communicate with a node that is uncompromised and has a closer match to the destination node compared to the Hamming distance of their IDs [31], where the Hamming distance is defined as a measure of the difference between two binary sequences of equal length. If there are no compromised nodes in the WSN, this scheme will always work as long as any two nodes can communicate. The following scheme can be run multiple times to increase the probability of success in generating a successful communication path between nodes.

First the source node generates a random number *r* and a counter *c* that initially starts at 0. Each round the source node increments *c* and computes the variable K_c, where $K_c = F(r,c)$ [31]. Next the source node generates a message *M*. This message contains the source and destination node IDs, K_c, *c* and a flag variable. The flag variable is used to govern the length of the path discovered and the number of messages sent [31].

Assume that another sensor node, *u*, receives the message *m*. This node initially tries to find a non-compromised node, node *v*, which it can compute a direct key. If this is successful, node *u* sets the flag variable in *m* to 1 [31]. It then sends the message to node *v*. If node *u* is unable to find a node that meets these requirements, and the flag variable in the message is set to 0, the path discovery stops [31]. On the other hand, it picks a non-compromised node *v* in which it can compute a direct key with. This path discovery continues until it reached the destination node *d*. When node *d* receives the key request, it sets the pairwise key as $K_{S,D} = K_c$ and tell the source node *s* the value of *c*. Finally, they two nodes *s* and *d* share a pairwise key [31].

(2) Performance and overhead for the hypercube scheme: In this hypercube key predistribution scheme, nodes accommodate *n* number of polynomials in their local memory. Each polynomial is shared by about *m* number of nodes in the network. Nodes can therefore establish $n(m-1)$ direct keys with other nodes [31]. The probability that two nodes can create a direct pairwise key decreases if the network size or the number of nodes in the network increases. This is offset by the fact that if there are no compromised nodes in the network, this scheme is guaranteed to establish a pairwise key between any two nodes either directly or indirectly [31].

Nodes in the hypercube key management need to store *n* polynomials and *t* number of compromised node IDs. The total storage overhead in this predistribution scheme is at most $n(t+1)log\ q + ntl$ bits, where $1 = \text{ceil}\ (log_2 m)$ [31].

There is no communication overhead if the two nodes can compute a direct key. In computational overhead, a node has to do $2(L+1)$ polynomial evaluations where $L =$ length of the key path. The first evaluation is the encryption, and the second evaluation is the decryption process in message communication [31].

(3) Security evaluation for the hypercube scheme: According to [31], there are two different attacks an adversary can launch against the hypercube key distribution scheme. The first is an attack on the pairwise key itself. The second is an attack on the entire network. The second attack attempts to lower the probability that two nodes can achieve a pairwise key altogether.

The first attack is an attempt to compromise the polynomials used in key generation between nodes *a* and *b* without compromising the nodes themselves. To accomplish this, the attacker must first compromise $t+1$ other sensor nodes. If the nodes *a* and *b* have computed an indirect key, the attacker must compromise the nodes used in the path detection that established the key. In total, the attacker must compromise $n \times (t+1)$ sensor nodes to effectively prevent nodes *a* and *b* from communicating with each other [31].

The second attack against this scheme is an attack against the network as a whole. One way to do this is to compromise a number, *b*, of the polynomials distributed to the nodes in the network. This will affect the indirect keys computed. Another way to attack the network as a whole is to randomly compromise individual sensor nodes. This could compromise the path discovery process and make it more expensive to create pairwise keys [31].

4.9. Key management schemes using deployment knowledge

Throughout the discussion in this paper, a significant piece of information regarding networks has not yet been mentioned, i.e., the deployment knowledge of these networks. As sensor nodes are randomly deployed in an area, it is difficult to obtain deployment knowledge. Some information on deployment knowledge is achievable if deployment followed a particular order. For example, if sensor nodes are scattered using an airplane pattern, these nodes might be grouped or placed in a particular order before deployment and, based on this pattern, an approximate knowledge of node positions can be acquired.

Deployment knowledge offers numerous advantages when used in networks such as achieving better storage, better resilience to node capture and more. In their study of key establishment techniques in sensor networks, Du, Deng, Han, Chen, and Varshney [8] propose a scheme using deployment knowledge that is based on the Basic Scheme [1]. Deployment knowledge in this scheme is modeled using probability density functions (pdfs). All the schemes discussed until now considered the pdf to be uniform; and when uniform, knowledge about the nodes can not be derived from it. Du et al. [8] consider non-uniform pdfs, which means that they assume the positions of sensor nodes to be at certain areas. Their method first models node deployment knowledge in a network and then develops a key predistribution scheme based on this model.

(1) Modeling of the deployment knowledge: The *deployment point* and the *resident point* are two terms that must be briefly understood when discussing the deployment model. *deployment point* of a sensor node is the point at which the sensor node is actually deployed; i.e., the node is dropped where the deployment is done through an airplane deployment point. *resident point* is the point at which the sensor actually resides after deployment. Let us assume the deployment area to be a 2-dimensional region $X \times Y$. The pdf for node *I*, for $I = 1, \ldots, N$ over the two-dimensional area is found by $f_I(x,y)$, where $x \in [0,X]$ and $y \in [0,Y]$. Generally nodes are deployed in groups, therefore the pdfs of the final resident points of all the sensors in a group is the same as the group of sensors deployed in a single deployment point. The group deployment model is designed as following in Du et al. [8]:

- *N* sensor nodes that are deployed in a place are divided into $t \times n$ equal size groups. Each group $G_{i,j}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, n$ is from the deployment point with index (i,j); and (x_i,y_j) is the deployment point for this group.
- The Deployment Model follows a grid-based approach with all deployment points arranged in a grid.
- The pdf of the resident points for node *K* in group $G_{i,j}$ is $f_K^{ij}(x, y|K \in G_{i,j}) = f(x - x_i, y - y_i)$.

Two groups that are deployed close together share some common keys. The amount of key overlap decreases as the deployment distance between the groups increases. When using the Basic Scheme, keys are drawn from the same key pool *S*; but, using the Deployment Model, different key pools are allowed for different groups so that the key pool can be divided into sub-key pools of $|S_c|$ keys each. The combination of all the sub-key pools still yields *S*.

Sensor nodes can be deployed in many different ways such as deployment through an airplane, using a vehicle, etc. In this scheme, deployment is considered as a Gaussian distribution, which is widely studied and practiced. The deployment distribution for any node *k* in group $G_{i,j}$ follows a two dimensional Gaussian distribution. The pdf of the resident points for the node *k* in group $G_{i,j}$ is [22] $f_k^{ij}(x, y|k \in G_{i,j}) = f(x - x_i, y - y_j)$.

When $f(x,y)$ is uniform, we cannot determine which nodes are close together prior deployment as the resident points of the nodes are uniformly distributed over the region. When $f(x,y)$ is random we can tell which nodes are close together. Though the distribution function is not uniform, the sensor nodes still need to be deployed evenly through the entire region. By selecting an appropriate distance between deployment points, the probability of finding a node in each small region can be made equal.

(2) Key predistribution using deployment knowledge: In a key predistribution scheme based on the Deployment Model, it is assumed that *N* sensor nodes are deployed in a place (point) and are divided into $t \times n$ equal size groups, each group $G_{i,j}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, n$. It is also assumed that the deployment points are arranged in a grid.

As with the Basic Scheme, key predistribution in the Deployment Model also consists of three phases: key predistribution, shared key discovery, and path key establishment. This scheme differs only in the first stage while the other two stages are similar to that of the basic scheme.

- *Key predistribution*: The most important step in this phase is to divide the key pool into $t \times n$ key pools. The goal of dividing the key pools is to ensure that neighboring key pools have more keys in common. Two key pools are neighbors if their deployment groups have nearby resident points. After the key pool is divided, each node in a group is selected and keys are installed from corresponding subset key pools.

- *Shared discovery phase*: In this phase each node must find its common keys shared with neighbors. There are many ways for doing this, but the simplest is to make the nodes broadcast their identifiers list to other nodes. If the nodes discover that they share a common key with other nodes, this key can be used as their communication link. When disclosing the nodes' identities is not desired, Merkle's Challenge Response Technique can be employed [9] in which each node sends a puzzle to neighboring nodes for each stored key and, if those nodes share a key in common with the source node, they will respond with the correct solution creating a key link for secure communication.
- *Path key establishment*: When two neighboring nodes do not share a common key, they can discover one using Path Key Establishment. If node U wants to communicate with node V and the two do not share a common key, node U must communicate with its neighbor node I, saying that it wants to communicate with node V. Node U then sends its ID and a secret key to node node I and, if node I shares a common key with node V, it sends the message to node V encrypted with that shared key. Through this path U $\rightarrow$ I $\rightarrow$ V or V $\rightarrow$ I $\rightarrow$ U, both nodes U and V can communicate with each other using a secret key.

(3) Creating key pools: Key pools that are deployed nearby should share certain keys in common. To assign keys to each key pool $S_{i,j}$ for $i = 1,\ldots,t$ and $j = 1,\ldots,n$, it is assumed that the pools are deployed in a grid: (a) key pools that are horizontal or vertical share $a|S_c|$ keys, where $0 \leqslant \alpha \leqslant 0.25$; (b) Key pools that are diagonal share $b|S_c|$ keys, where $0 \leqslant b \leqslant 0.25$ and $4a + 4b = 1$; (c)Two non-neighboring key pools share no keys.

Here, (a) and (b) are overlapping steps and to achieve the properties stated, the key pool is divided into eight total partitions, each with keys that are shared by the other nodes. Du et al. [8] developed a method to select keys for each subset key pool $S_{i,j}$, considering a grid scheme and given a global key pool S (the subset of key pools for each deployment point). The keys for the first subgroup (the group placed in the first row and first column) $S_{1,1}$ are selected from the global key pool S, and then keys for the second group in the same row are selected from the row left to it and S. This process continues for each row from left to right:

- Select $|S_c|$ keys for the group placed in $S_{1,1}$ and remove those keys from S;
- Select $a|S_c|$ keys for group $S_{1,2}$ from the key pool $S_{1,1}$, and the remaining keys w from the global key pool S, and then remove the selected w keys from S.
- Select $a|S_c|$ keys for group $S_{2,1}$ from the key pools $S_{1,1}$,$S_{3,1}$,$S_{2,2}$ and select $b|S_c|$ from the key pools $S_{1,2}$ and $S_{3,2}$. Then select and remove the remaining w keys from the global key pool S.

If w is the remaining keys that are to be selected from S, and $S_{i,j}$ be the group number in a grid, the selection procedure for different groups will be:

$$W = \begin{cases} [1-(a+b)] \times |S_c|, \text{for } j = 1 \\ [1-2(a+b)] \times |S_c|, \text{for } 2 < j \leqslant n-1 \\ [1-(2a+b)] \times |S_c|, \text{for } j = n \end{cases}.$$

Selecting the size of the key pool $|S_c|$ from S is also critical in this scheme. To ensure that no key in the network is shared between more than two nodes, a rule must be established: if a group G_1 selects the required keys from its neighbor G_2, no other group is allowed to select those keys. Since each group selects distinct keys from their neighbors, the size of $|S_c|$ is equal to the sum of all those keys and is calculated as $|S_c| = \frac{|S|}{tn-(2tn-t-n)a-2(tn-t-n+1)b}$.

In their paper, Du et al. [8] analyzed the performance of their scheme based upon the connectivity, communication overhead, and resilience to node capture of the network. Two parameters for connectivity called the Global Connectivity and Local Connectivity are defined for analyzing the performance of connectivity in the Deployment Model. Global Connectivity is the ratio between the nodes that are isolated and the total number of nodes in the network. Local Connectivity refers to the probability of any two nodes sharing at least one key. Both Global and Local Connectivity are affected by the key predistribution scheme where $B(n_i,n_j)$ is the event that two nodes share at least one key, and $A(n_i,n_j)$ is the event that both the nodes are neighbors, hence $p_{local} = \Pr(B(n_i,n_j)|A(n_i,n_j))$.

Let λ be the ratio of the shared key pool between two nodes and $|S_c|$. Du et al. [8] calculated the probability $p(\lambda)$ that two nodes share at least one key when they have $\lambda|S_c|$ in common as $p(\lambda) = 1 - \Pr$(two nodes do not share any key).

Global Connectivity is also required for better efficiency. The key sharing in a wireless network the using Deployment Model is not uniform, so uniform distribution techniques like Erdos' Random Graph Theory cannot be applied. Shakkottai et al. [23] determined the connectivity of a wireless network with unreliable nodes that can be used for random distribution. The authors [8] simulated the results for Global Connectivity using m for the number of nodes in a node's key ring, which showed that when $m = 200$, no nodes are wasted due to a lack of security links while only 0.12 percent of nodes are wasted when $m = 100$. The overlapping factors a and b has to be selected carefully, when $a = 0.25$ and $b = 0$, each group shares keys only with horizontal and vertical neighbors only; when $a = 0$ and $b = 0.25$, each group shares keys only with diagonal neighbors only. The values of a and b depend upon many factors like the size of the nodes key ring m, different types of neighbors, etc.

Advantages of this scheme include that only to consider deployment knowledge that can minimize the number of keys and help increase resilience or resistance to node capture and reduce network overhead, it increases overall connectivity of the network graph, and offers same pros as the Basic Scheme on which it is based. Disadvantages of this scheme include complexity.

4.10. Location dependent key management scheme

Most key management schemes in sensor networks do not consider locations of the sensor nodes after they are deployed into the environment. A location dependent key management scheme proposed in [32] decides which keys to put on each node depending on their locations in the environment.

In this scheme, nodes are determined to be static. They communicate only through encrypted channels, and nodes can be added at any time. Also nodes in this scheme are assumed to be capable of transmitting at different power levels giving different transmission ranges. Also there exist special nodes called anchors. The only difference between the anchor nodes and the other nodes in the network is that the anchor nodes transmit at different power levels and they are tamper proof [32]. There are fewer anchor nodes than normal nodes in the network. There are N_s sensor nodes and N_a anchor nodes. There are three phases to this type of WSN. They are the predistribution phase, initialization phase, and communication phase [32]. In the predistribution phase, a key server computes a set of keys to be used by the nodes. It places the keys into a key pool. Each sensor node is then loaded with a subset of these keys along with a single common key every node shares. Anchor nodes do not get keys from the key pool.

The next two phases occur after the nodes have been deployed into the environment. All of the nodes and anchors are randomly distributed. The anchor nodes now transmit a beacon at different power levels. The sensor nodes receive these beacons and compute new keys using their old keys and the beacon received from the anchor node. The original subset of keys is deleted from the memory of the sensor node after they compute their new keys. Also, all of the sensors, except the anchor nodes, delete the one common key they all share [32]. When this is finished, the initialization phase is complete. When a node needs to update its keys, it follows this pattern again. Since it computes its keys from an anchor node that is close to its location, the keys that are in the sensors memory are directly connected with the sensors' physical locations in the network [32].

Next is the communication phase. In this phase, the nodes compute pairwise keys to establish secure communication among them.

One of the key advantages of this location aware key management scheme is that compromised nodes do not affect nodes in a different location in the network. If an adversary compromises a node, they are unable to communicate with other nodes that are attached to different anchor nodes. Therefore compromised nodes only affect other local nodes, but not the entire network [32]. This is because keys in different physical locations are generated by other anchor nodes and are therefore different.

(1) Performance of the location dependent scheme: The impacts of different power levels, common key thresholds, anchor transmission radius, and sensor node transmission radius are examined in [32].

The first evaluation is the impact of the number of power levels on anchor nodes. The number of keys in the key pool, P, and the number of keys on each node, R, are chosen to yield a high connectivity ratio. When the P/R ratio is increased, the impact of a compromised node is decreased [32].

The next evaluation is the impact of the common key threshold N_c. If the common key threshold N_c is increased, the connectivity ratio decreases when R is low.

The anchor transmission range also has an impact on performance of this scheme. If the maximum range of the anchor node is increased, initially the compromise ratio decreases, and then it increases as the range of the anchor node grows [32]. This is caused by the fact that if the anchor nodes transmission range was large enough to cover the entire network, all nodes would have beacons from all anchor nodes and keys would not be as diverse.

The sensor transmission range is last to be evaluated. It is shown in [32] that the connectivity ratio and compromise ratio decreases when the sensor nodes transmission range is larger than the anchor node transmission range.

(2) Location dependent schemes versus other schemes: When P and R equal 1, this is the worst case for the location dependent scheme since an adversary only has to compromise one node to gain access to the entire network. This scheme also performs worse than a random key distribution scheme where the other scheme has a key pool of 5000 and 175 keys on each node [32]. As the numbers of nodes in the network that are compromised are increased, the performance of the random key distribution scheme deteriorated faster than the location dependent scheme [32].

In the location dependent key management scheme, an adversary can launch a denial of service attack if they jam the anchor nodes and transmit false beacons. This is fairly hard to accomplish since anchor nodes are randomly dispersed in the environment [32].

Location dependent key management schemes can have a positive effect on the security of the network. It can also increase the direct connectivity ratio of neighboring nodes. By knowing where nodes reside in the network, keys can be generated for specific areas. This quarantine node compromises because keys for different areas of the network will be different. Also since the location of nodes are known, the keys generated for each area allow neighboring nodes to communicate with each other with higher probability.

4.11. Location aware combinatorial key management

The authors in [33] proposes a lightweight combinatorial construction of the key management scheme for clustered WSNs, called SHELL. In SHELL, collusion is reduced by using the nodes' physical locations in computing their keys. This scheme uses a command node to govern the entire network. The command node directly communicates with the gateway nodes which are in charge of the individual clusters. Nodes can be added to this network at any time. The gateway nodes are powerful enough to communicate with the command node and do the required key management functions.

(*1*) *Threat model:* The threat model is assumed to be an adversary that tries to capture and compromise a number of the nodes in the network. There also is no unconditional trust on any sensor node. If an adversary compromises a node, the memory of that node is known to them. Gateway nodes can also be compromised. The goal of the adversary is to uncover the keys used in the network for secure communication [33]. To aid the adversary in doing this, they attempt to get nodes to collude with each other.

(*2*) *SHELL key management scheme:* Each gateway node can communicate with at least two other gateway nodes in the network, and has three types of keys [33]. The first key type is a preloaded key that allows the gateway to directly communicate with the command node. The second type allows the different gateway nodes to communicate. The third key type allows the gateway to communicate with all of the sensor nodes in its cluster.

The command node is assumed that it is unable to be compromised [33]. Some of the responsibilities of the command node are to act as a key repository, authenticate gateway and sensor nodes, distribute keys to all other nodes, and it performs key renewal when needed.

The first thing that happens after deployment is that the gateway nodes establish a connection back to the command node. When the command node has made connections with all gateway nodes, it then computes link specific keys for inter-gateway communication [33]. The command node then sends these keys to each of the gateway nodes. The command node also sends the gateway nodes the keys to use for communication with the sensor nodes in their particular clusters. When all of the gateway nodes have been established and linked, they begin inter-cluster sensor discovery.

The gateway node that governs a particular cluster does not generate the keys for its cluster. The command node assigns two other cluster heads to generate keys for this cluster. Once the keys are computed, each node is notified about the set of administrative keys that it was assigned [33]. The cluster heads then transmit the keys that were generated to the sensor nodes in their cluster.

When this is finished, the network bootstrapping phase is over. This bootstrap is run every time rekeying is necessary or when new nodes are added to the network [33].

Sensor nodes can be added to the network at any time. When a new node is added to the network, the command node lets the gateway node to broaden its power range and gives it the key to communicate with the new node [33]. The gateway nodes decide among themselves on which cluster the new node will join.

(*3*) *Attacks on the SHELL scheme:* If nodal compromise or fault is detected, a key revocation is initialized. The gateway node watches the nodes in its cluster for failure or compromise and the command node does the same thing for the gateway nodes [33].

If the gateway is compromised, the command node is notified. The command node then assumes the responsibility of initializing a rekeying of the intergateway nodes. There are two ways of accommodating for a compromised gateway node [33]. The first is to deploy a new gateway node. The other way is to distribute the nodes headed by the compromised gateway node to other clusters. If the option to replace the gateway node is used, and the nodes in its governed cluster cannot communicate with it, a key redistribution is required.

In the case a sensor node is compromised, it is given that the gateway node can detect it. If a single sensor in a cluster is compromised, all of the senor nodes keys have to be replaced. This is done to protect the rest of the network for the adversary [33].

In some key management schemes if a few nodes collude, the adversary can gain knowledge about the entire network. If this happens this is considered as a network capture. SHELL attempts to correct this problem by increasing the number of nodes that need to collude in order to reveal information about the network [33]. In order for any two or more sensor nodes to collude, they must be within transmission range of each other. This is most likely to happen when a neighboring node has been compromised and the adversary can manipulate this node. When two nodes collude, they both know all keys that are common to each other [33]. Nodes need to be in direct transmission range of each other to collude.

5. Dynamic key management

In [30], Eltoweissy et al. propose a dynamic key management system, called exclusion-based system (EBS). Some of the advantages of using a dynamic key management scheme are improved network survivability and better support for network growth [30]. The issue in creating a dynamic key management system is being able to make it secure and efficient.

The EBS assigns each node k keys from a key pool of size $k+m$. If node capture is detected, rekeying occurs throughout the network. A disadvantage to this EBS scheme is that if even a small number of nodes in the network are compromised, information about the entire network could be uncovered by an adversary.

The first application of the EBS scheme was done with anonymous nodes in the network. The nodes did not have

IDs [30]. Instead, nodes were identified by their locations. This scheme is heterogeneous and depends on a central base station for key distribution. This EBS scheme is very efficient, but it does not prevent collusion among nodes that are compromised.

Since there is a collusion problem in the standard EBS system, a new proposed system called SHELL [30] uses node location information to compute keys. Clusters and gateways are used. The clusters in this scheme track key assignments but not the keys themselves [30]. The actual keys are stored in the gateways of other clusters. This system is collusion-resilient. SHELL gathers node locations after employment and uses this information while assigning keys. Nodes that are located closer to each other share a higher number of keys than nodes that are located longer distance from each other [30].

LOCK, localized combinatorial keying, is a dynamic key management scheme based on the EBS scheme [30]. This wireless sensor network model consists of a three level hierarchy, base station, cluster heads, and sensor nodes. LOCK does not use location information in the generation of keys. When the nodes are initially released into the environment, they create a set of backup keys. These sets of backup keys are only shared with the base station, not the local cluster leader nodes. If a node is captured, other nodes are rekeyed locally so that the compromised node is unable to communicate with them. If a cluster leader is compromised, the base station initiates a rekeying on at the cluster head level. Also, nodes within the group governed by the compromised cluster leader rekey with the base station [30]. In LOCK, if an adversary compromises any node, it does not have an effect on the operations of other nodes in other clusters.

Static key management schemes rely on predistribution of the keys and the probabilistic chance that two nodes that want to communicate share at least one of these keys. These static key schemes typically have a large value of keys (250 or higher according to [30]) to ensure a high probability that nodes can communicate. In some cases though, the storage capacity of the sensor nodes is not be able to hold a very large number of keys. The keys that each node gets before distribution, k, is randomly chosen from a key pool, m. m needs to be large so the number of nodes an adversary would need to compromise to gain information about the entire network has to be large. On the other hand, if m gets very large, the probability that two nodes receive a common key is lowered [30]. Dynamic schemes work in generally the same way, but with lower values of m [30]. This guarantees the dynamic schemes a higher probability of connectivity.

The static and dynamic schemes both use random assignment of keys to nodes. Static schemes just use a larger k values from a larger key pool. The dynamic schemes use smaller k values on the nodes and a small key pool [30].

Some of the differences in static and dynamic key schemes are now examined. Static schemes typically have short-lived network life, larger key pools and keys in local nodes, and may have a large rekeying cost since the key pool is large. Dynamic schemes usually have longer network life spans, smaller key pools and fewer keys on each node, and only require a few messages for rekeying [30].

6. Hierarchical key management

6.1. LEAP: Efficient security for large-scale WSNs

One of the important mechanisms in sensor networks, in-network processing, is not considered in the previous schemes. This critical issue must be handled while dealing with the resource-constrained property of WSNs. Most of the data has to be collected by an aggregator node and then passed on to other nodes in a WSN; however, data fusion through in-network processing can be used to save network energy and reduce communication overhead. The key establishment techniques that have been discussed so far do not support an In-Network Processing approach because the nodes in this method are unable to communicate with each other before transmitting data. Passive Participation is a form of In-Network Processing in which a sensor node takes certain actions based on messages from other nodes. Zhu, Setia, and Jajodia [6] devised a scheme called LEAP that would allow for data fusion, In-Network Processing and Passive Participation.

Besides offering basic requirements like confidentiality and authentication, LEAP supports various communication patterns, including unicast (addressing a single node), local broadcast (addressing a group of nodes in a neighborhood), and global broadcast (addressing all the nodes in a WSN). Sometimes WSNs are deployed in an adversary's arena and, where most of the time compromised nodes are undetected, LEAP provides survivability such that compromising of some nodes does not cede the entire network. LEAP is energy efficient since it supports techniques like In-network Processing and Passive Participation that greatly reduce network communication overhead and, in turn, increase node battery life. Furthermore, LEAP ensures that messages transferred are not fragmented, which would increase packet losses in transmission as well as make protocol implementation more complex and difficult.

LEAP is based on the theory that different types of messages exchanged between nodes need to satisfy different security requirements. All the packets transferred in a sensor network need to always be authenticated where a sensor node knows the sender of the data since an adversary may attack a WSN with false data at any time. On the other hand, confidentiality, like encryption of packets carrying routing information, is not always needed. Different keying mechanisms are necessary to handle the different types of packets. For this, Zhu, Setia, and Jajodia [6] establish LEAP with four types of keys that must be stored in each sensor: individual, pairwise, cluster, and group. Each key has its own significance while transferring messages from one node to another in a WSN and by using these keys, LEAP offers efficiency and security with resistance to copious attacks such as the worm hole and the sybil.

- *Individual key*: This is a unique key that is shared between the base station and each sensor node. Sensor nodes use this key to calculate the MACs on their messages to the base station like alert signals (reports on abnormal nodes). In the same way, a base station can use an individual key to send messages to each and every node in the network.
- *Pairwise shared key*: This is a unique key that is shared between each node and its neighboring node in the network. A node can use it to transfer individual messages like sharing a cluster key or sending data to an aggregator node.
- *Cluster key*: This is a key that is shared between a node and its neighboring nodes, and is very important since it supports In-network processing and passive participation. A node may elect not to send a message to the base station if its neighboring node is sending the same message with a better signal, a discovery that is only possible to implement if a node shares a common key with its neighboring nodes. With such a cluster key, a node can select which messages to transfer, thereby reducing the system communication overhead.
- *Group key*: The base station shares this key with all the nodes in the network to send queries to them. Group key used requires an efficient rekeying mechanism for updating it as there is a chance for an adversary to know the key whenever a node is compromised.

(1) Efficiently establishing LEAP [6]*: Establishing individual keys:* Every node in a WSN shares a unique key with the base station that is preloaded into each node's memory before being deployed. The individual key K_u^m for node U is calculated as $K_u^m = f_{K^m}(u)$. For this, f is a pseudo-random function and K^m is the master key known only to the controller. There is no need for the base station to store all the individual keys, because the base station generates them on the fly whenever it attempts to communicate with a node.

Establishing pairwise shared keys: The most common key used in a WSN is the pairwise that is shared between each node and its neighboring node. Sensor nodes are randomly scattered in an area; therefore, the key establishment technique used should guarantee that nodes discover neighboring nodes when deployed. Because sensor nodes are static, the key establishment technique does not have to consider deployment knowledge of others before node deployment. When an adversary obtains a sensor node, it is assumed that the node cannot be compromised before time $t_{\min}$. Whenever a node is deployed in a WSN, it requires some minimum time to identify neighbors and establish keys with them, which will be t_{est}. It is expected that $t_{\min} > t_{\text{est}}$; otherwise, the adversary could easily capture all the nodes in the WSN and effectively take over the entire system.

The process of establishing keys when nodes are already deployed is similar to the process of key establishment when a new node is added to the network. There are four stages that represent the key establishment of new node U deployed in the network: key predistribution, neighbor discovery, pairwise key establishment, and key erasure. During the initial stage of key predistribution, node U is loaded with the key K_i by the controller and derives the master key K_u using it. For neighbor discovery, node U first initializes a timer to activate at time $t_{\min}$, then starts communicating with its neighbors by broadcasting a HELLO message containing its ID. Node V responds to this message with a reply containing its ID. The ACK of V is then authenticated using its master key K_v derived from K_i. Node U verifies the authentication of V by generating the master key K_v as node V shares K_i with it: $U \to *{:}U$ and $V \to U{:}V{,}MAC(K_v,\ U|V)$.

For the third stage of pairwise key establishment, node U computes the pairwise key K_{uv} with node V using V's identity. Node V can also do the same thing with U. There is no need for authenticating node U to V as any future messages authenticated with K_{uv} will prove node U's identity. In the fourth and final stage, key erasure, node U erases K_i and all the master keys of the other nodes after the time expires. Then node U will not be able to establish pairwise keys with any other nodes in the WSN so that, though an adversary captures a node, the communications between it and another node cannot be decrypted without the key K_i.

Pairwise shared keys do have computational overhead since each node U in the network must verify the MACs generated by neighboring nodes and each must reply with a message including its identity and an MAC. Each node must also generate a pairwise key between every other neighboring node in the network. Because a HELLO message includes only a node ID and an *ACK* message has only an ID and an *MAC*, both can be adjusted in a single packet. Also the space required for storing a preloaded is only one key K_i; therefore, the communication and storage overheads are small.

The HELLO message in our scheme is not authenticated, so an adversary may try to attack the network by constantly sending these messages, which will drain a WSN's resources. There are two solutions for this attack: the controller may try to load each new node with the group key of the network so that the nodes can verify the authentication of the message by verifying the group key in the message, or else the controller might try to add some randomness into the IDs of the newly added nodes such that false ones will be detected and dropped. The assumption made here is that the sensor nodes are able to permanently eliminate the master key K_i from their memory, which may not be possible in all cases. One of the unique advantages of the scheme above is that once pairwise keys are established between neighboring nodes in an area of a WSN, they cannot be established again, which protects the network from clone attacks. In clone attacks, an adversary tries to attack the network by installing a number of nodes with keys acquired from compromised nodes, which then establish pairwise keys with other nodes in the network and compromises the entire WSN with just a few nodes. The scheme stated above restricts this kind of attack to a local area as the cloned nodes cannot establish pairwise keys with other nodes in the network that are not neighboring nodes of the one compromised.

The security of the above scheme can be even increased by regularly changing the master key K_i. If an adversary compromises the sensor node before the establishment time, the master key K_i can be obtained and then the whole network can be compromised. By changing the master key regularly, not only is this attack averted, but also attacks caused later by the same adversary are averted, as the master key could still be acquired by compromising a node and deriving the key from its memory. There is one more security threat that must be addressed here: if an adversary attacks the WSN and succeeds in compromising the nodes before key establishment time t_{est}, the network can then be attacked through new nodes added into the network using the correct master key. This problem, however, can be solved. Suppose that the controller wants to add N_i nodes into the network in the time interval T_i, N_i ID's are generated for the nodes based on a random seed S_i and each of the N_i nodes is loaded with a unique ID. The nodes can now establish pairwise keys as stated and when the controller later broadcasts N_i and S_i into the network using a broadcasting scheme like μTESLA, the nodes verify whether those attempting communications are valid or not based on N_i and S_i. The pairwise keys of nodes that are not valid are then deleted from the memory of all the nodes. Compared to other approaches, this method offers greater efficiency and smaller overhead, while also protecting the network from clone attacks and other serious attacks.

Establishing cluster keys: The cluster key establishment is based on the pairwise key establishment. If node U wants to establish a cluster key with its neighbors $v_1, v_2, v_3, \ldots, v_n$, first it generates a key K_c and then encrypts that key using the pairwise key which it shares with each neighbor. Node U then transmits this encrypted message to its neighbors. Node v_1 decrypts the key using the pairwise key which it shares with U, and then stores the key in a buffer. Next it sends back its own cluster key to node U. When any of the nodes are revoked, node U generates a new cluster key in the same way and transmits the key to all remaining nodes.

Establishing group keys: A group key, which is shared between a base station and all the nodes in a WSN, is needed when the base station wants to send a message or query to all the nodes of that WSN. One way of achieving this is using the hop-by-hop method in which the base station encrypts messages using the cluster key which it has and then broadcasts the message to all the nodes in its neighborhood. The nodes would decrypt the message and then encrypt it using the cluster key which they share with their neighbors. In this way, the message can be received by all the nodes in the network. This is efficient, but has an overhead of encryption and decryption at every node.

A simple method to establish a group key is to preload each node with the group key before deployment, but this is still within the scope for rekeying the group key which will be necessary. Unicast-based group rekeying can also be considered for which the base station needs to send the group key to each node in the network, but this involves much communication overhead. However, Zhu, Setia, and Jajodia [6] proposed an efficient scheme based on cluster keys in which the transmission cost will only be one key. In WSNs, all messages sent by the base station must be authenticated or an adversary may impersonate it. The group key must be updated every time when a node is revoked. Therefore the first issue to consider is how node revocation can be done in this scheme. μTESLA, based on a one-way key chain and delayed disclosure of keys, is an efficient method to broadcast messages into a WSN, as previously discussed. To bootstrap μTESLA, each node should be preloaded with the commitment of the key chain. If K_g is the new group key and U is the node to be revoked, the base station broadcasts the following message: $M : \text{Controller} \rightarrow * : u, f_{K'_g}(0), MAC(k_i^T, u|f_{K'_g}(0))$, where $f_{K'_g}(0)$ is the key that enables the node to verify the authentication of the group key. The server then distributes the MAC key k_i^T after one μTESLA interval. After a node V receives the message M, it verifies the authenticity of the message using μTESLA. If node V is neighbor of U, V will remove its pairwise key shared with U and update its cluster key. This process can also be used for updating the group key.

For secure key distribution, this scheme uses a protocol that is the same as the beaconing protocol for which all the nodes are organized in a breadth-first spanning tree where each node not only remembers the parent and children of a spanning tree, but also the other neighbors. The new group key K'_g is distributed to all the nodes in the network using the spanning tree established by the routing protocol. The base station initiates the process by sending the group key to all its neighbors in the network. The nodes that receive the message verify its authentication by calculating $f_{K'_g}$ and by checking whether it is the same as the verification key received earlier in the node revocation message. The algorithm continues recursively down the spanning tree with the help of each node, which transmit the group key to neighbors while encrypting the message using their cluster keys. As discussed earlier, this hop-by-hop scheme does not involve much overhead as only one key is encrypted and decrypted, and as rekeying of the group key is infrequent. However, it is desirable to change the group key more often or an intruder may compromise the entire WSN by obtaining one node and, thereby, deriving the group key.

(2) Local broadcast authentication: Local broadcast is different from global broadcast in that in local broadcast a node generally does not know what packet it is going

to generate next and messages generally consist of aggregated sensor readings or routing protocols. μTESLA is not suitable for local broadcasting because μTESLA does not provide authentication immediately, which is needed in some local broadcast cases. Also, in μTESLA nodes need to keep the packets in their buffers until the authenticating key arrives, which increases the storage space required. A packet that has to travel L nodes will at least need L μTESLA intervals, thereby affecting latency of the network. Pairwise keys cannot be used for local broadcast because, if a node has n neighbors, the approach requires the sender node to calculate n *MACs* for each message. Local broadcast needs a method where a node can broadcast a message to all its neighbors using a single *MAC* and cluster keys, with a problem as follows. If an adversary can compromise a node, the cluster key from that node is available and can be used to attack the network by impersonating that node or a neighboring node. If nodes X, U, and V are three vertices of a triangle, X is compromised, and U wants to send messages to X and V, X can use node U's cluster key to impersonate it and send false messages to V.

Fortunately, Zhu, Setia, and Jajodia [6] have designed a scheme called One-Way Key Chain-Based Authentication for defeating this attack totally. This scheme is based on μTESLA in that each node generates a one-way key chain and sends the commitment of it to their neighbors. This transferring is done using the pairwise keys already shared with neighbors. If a node wants to send a message to its neighbors, it attaches the next authorization key from its key chain to the message. The receiving node can verify the validation of the key based on the commitment it has already received. The One-Way Key Chain-Based Authentication is designed based on two observations: a node only needs to authenticate to its neighbors and that a node V will receive a packet before a neighboring X receives it and resends it to V. This observation is true because of the triangular inequality among the distances of nodes involved. An adversary may still try to attack the nodes by shielding node V while U is transmitting a message, and then later send a modified packet to V with the same authorization key; but this attack can be prevented by combining the authorization keys with the cluster keys. When this is done, the adversary does not have the cluster key and so cannot impersonate node U. However, this scheme does not provide a solution for attacks from inside where the adversary knows U's cluster key.

(3) LEAP performance evaluation: Overhead: Since the performance of the pairwise key establishment have already been discussed, here we review other factors of performance like the computational cost, communication cost and storage requirement of this approach. As mentioned, a cluster key is established based upon the pairwise keys of a node. Let us suppose that the number of neighbors to a node is n; if the cluster key has to be updated the scheme must perform n encryptions, which is computationally expensive. The value of n depends upon the density of the scheme; the computational cost increases as the network is denser. While for securing distribution of a group key, the number of decryptions is equal to the size of the network. The total number of encryptions is also equal to the size of the network; so if the size of the network is M, the total number of symmetric operations will be $2M$. From these derivations, the computational cost of the scheme is dependent upon the density of the network d. Zhu, Setia, and Jajodia [6] stated that the average number of symmetric operations of the scheme is about $2(d-1)^2/(M-1)+2$. If the density of the network is reasonable, the computational cost may not be a bottleneck to the scheme.

Also, the cost decreases with the increases of M. The communication cost of the scheme is the same as the computational cost. The average number of keys a node has to transfer for updating keys due to revocation is $(d-1)^2/(N-1)+2$. Just like computational cost, communication cost increases with a increase in the density of the network and decreases with an increase in the size of the network. The storage requirement of this scheme is a bit high because each node must store four types of keys in it. Considering the degree of node to be d, a node has to store one individual key, d pairwise keys, d cluster keys, and one group key. Also, a node must store a one-way key chain and a commitment for each neighbor for local broadcast. If L is the number of keys stored in a key chain, the total number of keys the node has to store in this scheme will be $3d+2+L$. Again, the storage requirement of LEAP depends upon the density of the network.

Resilience to attack: An adversary might launch a selective forwarding attack in which a compromised node drops the packets containing the routing information of selected nodes and forwards the other packets normally. LEAP can minimize the affects of the scheme by minimizing this problem to a local area. As LEAP uses local broadcast, the attack's effects will not transfer to more than 2-hops, which will result in defeating the purpose of such an attack. LEAP can also prevent a HELLO attack in which an adversary attacks the network by repeatedly transmitting HELLO messages and thereby depletes the network's resources. This attack is averted since the nodes in a LEAP scheme accept packets only from authenticated neighbors. The sinkhole and wormhole attacks, however, are difficult to solve. In the sinkhole attack, a compromised node attracts packets by advertising information like high battery power, etc., then later drops all the packets. In the wormhole attack an adversary launches two nodes in the network, one near the target of interest and the other near the base station. The adversary then convinces the nodes near the target, which would generally be multiple hops away from the base station, that they are only two hops away thereby creating a sinkhole. Also, nodes that are far away think that they are neighbors because of the wormhole created. In LEAP an adversary cannot launch a wormhole attack after key establishment as at that point every node has knowledge about its neighbors so it is not easy to convince a node that it is near a particular compromised node. An insider node must then succeed in compromising two nodes for creating a wormhole and those nodes must be near the target of interest and the base station after the key establishment phase is complete. Although an adversary may try, it is difficult to create an attractive sinkhole without being detected.

LEAP includes efficient protocols for supporting four types of key schemes for different types of messages broadcasted in WSNs and includes an efficient scheme for local broadcast authentication. LEAP is an efficient scheme for key establishment that resists many types of attacks on the network, including the sybil, sinkhole, wormhole, and so on. LEAP also provides efficient schemes for node revocation and key updating in WSNs.

Advantages of this scheme include that it offers efficient protocols for supporting four types of key schemes for different types of messages broadcasted, reduces battery usage and communication overhead through In-Network Processing, and uses a variant of μTESLA to provide local broadcast authentication. Disadvantages of this scheme include that it requires excessive storage with each node storing four types of keys and a one-way key chain, computation and communication overhead dependant upon network density (the more dense a network, the more overhead it has).

6.2. Key managements for heterogeneous sensor networks

Previous research on sensor network security mainly considers homogeneous sensor networks. Research has shown that homogeneous ad hoc networks have poor performance and scalability. Furthermore, many security schemes designed for homogeneous sensor networks suffer from high communication overhead, computation overhead, and/or high storage requirement. Recently deployed sensor network systems are increasingly following heterogeneous designs. In [24,25], Du et al. considered key management in a Heterogeneous Sensor Network (HSN) that consists of a small number of powerful High-end sensors (H-sensors, e.g., PDAs) and a large number of Low-end sensors (L-sensors, e.g., the MICA2-DOT nodes).

In [24], Du et al. present an effective key management schemes – the asymmetric predistribution (AP) scheme for HSNs. The powerful H-sensors are utilized to provide simple, efficient and effective key set up schemes for L-sensors. Although tamper-resistant hardware is too expensive for L-sensors, it is reasonable to assume that powerful H-sensors are equipped with this technology. The basic idea of the AP key management scheme is to pre-load a large number of keys in each H-sensor while only pre-loads a small number of keys in each L-sensor. An H-sensor has much larger storage space than an L-sensor, and the keys pre-loaded in an H-sensor are protected by the tamper-resistant hardware. The performance and security analysis shows that the AP key management scheme can significantly reduce the sensor storage requirement while achieving better security (e.g., better resilience to node compromise attack) than several existing sensor network key management schemes.

In [25], Du et al. design an efficient key management scheme for HSNs by utilizing the special communication pattern in sensor networks. In most wireless sensor networks, the many-to-one traffic pattern dominates, where a large number of sensors send data to one (or a few) sink. Hence, a sensor node may only communicate with a small portion of its neighbors. Most existing sensor key management schemes try to establish shared keys for all pairs of neighbor sensors, no matter whether these nodes communicate with each other or not, and this causes large communication and computation overhead. In the paper, the authors adopt an HSN model for better performance and security. In [25], Du et al. propose a routing-driven key management scheme, which only establishes shared keys for neighbor sensors that communicate with each other. Elliptic Curve Cryptography is utilized to further increase the efficiency of the key management scheme. The performance evaluation and security analysis show that the routing-driven key management scheme provides better security with significant reductions on communication overhead, storage space and energy consumption than some existing sensor key management schemes.

6.3. Pairwise keys in heterogeneous sensor networks

Similar to [24,25], Traynor et al. [29] also assume that there are nodes in the network that are more powerful and more secure than others, and these more powerful nodes are also in tamper proof boxes or well guarded. A node that has limited memory and processing power is identified as L_1 and a node that have more memory and more processing power is identified as L_2 [29]. L_2 nodes act as head nodes for the L_1 nodes and have the responsibility of routing packets throughout the network. These L_2 nodes have access to gateway servers which are connected to a wired network.

In [29], there are three keying and trust models for the heterogeneous network. The first is called backhaul, in which the L_1 nodes only send data to L_2 nodes and only do this if they both directly share a key. The second scheme is called the peer-to-peer with limited trust, in which two L_1 nodes wish to exchange data and they only trust each other or another L_2 node. If the two L_1 nodes can establish a pairwise key, they communicate with each other. If they are unable to establish a key, they turn to the L_2 node for assistance in establishing one. The third scheme is peer-to-peer with liberal trust, which works like the second scheme, but instead of L_1 nodes' only trusting the other L_1 it wants to communicate with and other L_2 nodes, the L_1 nodes trust all other L_1 nodes and L_2 nodes in the network [29].

In the heterogeneous scheme, compromised nodes are unable to eavesdrop on messages other than the ones meant for it directly [29]. The following scenarios are examined under the peer-to-peer with liberal trust model. The first situation explored is the level 1 node compromise. Nodes in the sensor network are likely to be compromised, especially L_1 nodes in the heterogeneous networks. Compromised nodes in a level 1 compromise are limited to sending false data into the network or learning new keys that are generated. If the compromised node is highly connected with other nodes, the information on keys it receives is higher than if it were a node on the edge of the network [29]. Since compromised L_1 nodes want to establish secure connections to receive data from its neighbors, most likely it would have to go through an L_2 node for help with key generation. Since this is the case, the L_2 nodes can sense irregular keying patterns from suspicious L_1 nodes that may indicate a compromised node [29].

The second type of compromise is an L_2 node compromise. These nodes are usually placed in tamper proof cases, but no sensor node can be 100 percent resistant to compromise [29]. Since L_2 nodes hold far more keys in its memory than L_1 nodes, a compromised L_2 node may affect a large portion of the network.

Another threat to the wireless sensor network is the capture and pooling of keys from compromised L_1 nodes. If the adversary can capture a larger number of L_1 nodes and pool the keys from these nodes, they have a higher probability of generating keys between other un-compromised L_1 nodes [29].

7. Conclusion

Key management for Wireless Sensor Networks (WSNs) is a critical issue that has been addressed through many proposed schemes presented in various papers. This paper provides an overview of these techniques, each of which offers different advantages and disadvantages. A balance between the requirements and resources of a WSN determines which key management scheme should be employed. A WSN used in a battlefield demands more security than one used in places like shopping centers; also, the former can be made more costly but the later needs to be as cheap as possible. Decisions regarding the key management scheme to be used must be based on these requirements for efficiency.

The study of key management in wireless sensor networks still has abundant research opportunities in the future. As electrical systems become smaller, more powerful, and use less energy, the security restraints will become more complex. As for now, key management systems are a trade-off of performance and security to low overhead in memory usage and message transmissions [30]. Key management systems sole purpose is to supply secure communication in wireless sensor networks without producing much overhead.

More schemes should be developed to make efficient use of sensor nodes' limited resources. Greater emphasis should be given to the security in key management schemes, particularly as a majority of sensor node deployment is in hostile environments where providing strong security features is a must. Though receiving much attention recently, there are many problems to be addressed in WSNs such as finding the compromised nodes in a network, making good use of deployment knowledge, making nodes tamperproof without much overhead, decreasing the bootstrapping time required for the network, and so on. Future research should especially seek techniques for compromised node discovery and efficient methods to revoke compromised nodes.

Numerous lives can be saved in wars with the data collected by sensors, but WSNs in the near future will offer many surprises for humans as they come to be used in daily household matters like locking doors and switching off electronics, or controlling traffic in high-volume areas. Sensors installed in big malls and shopping centers can guide people to their required products easily while those in hospitals can monitor patient condition and those in forests can provide immediate knowledge about disastrous hazards like wildfire. These advantages are only a small fraction of what WSNs could potentially offer when deployed more commonly. Future studies can prove WSNs to be useful in a wider variety of environments.

References

[1] L. Eschenauer, V.D. Gligor, A key management scheme for distributed sensor networks, in: Proceedings of the 9th ACM Conference on Computer and Communication Security.

[2] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in: Proceedings of the 2003 IEEE Symposium on Security and Privacy, May 11–14, pp. 197– 213.

[3] D. Liu, P. Ning, Establishing pairwise keys in distributed sensor networks, Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03) (2003) 52–61.

[4] W. Du, J. Deng, Y.S. Han, P.K. Varshney, A pairwise key predistribution scheme for wireless sensor networks, Proceedings of the 10th ACM Conference on Computer and Communications (SecurityCCS'03) (2003) 42–51.

[5] A. Perrig et al., SPINS: security protocols for sensor networks, Proceedings of ACM MOBICOM (2001).

[6] S. Zhu, S. Setia, S. Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks, in: Proceedings of The 10th ACM Conference on Computer and Communications Security (CCS '03), Washington D.C., October, 2003.

[7] D. Carman, P. Kruus, B. Matt, Constraints and approaches for distributed sensor network security, NAI Labs Technical Report No. 00-010, September 2000.

[8] W. Du, J. Deng, Y.S. Han, S. Chen, P.K. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, in: Proceedings of IEEE INFOCOM 2004.

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

[10] R. Anderson, A. Perrig, Key infection: smart trust for smart dust, Unpublished Manuscript, November 2001.

[11] R. Merkle, Protocols for public key cryptosystems, Proceedings of 1980 IEEE Symposium on Security and Privacy (1980).

[12] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory IT-22 (1976) 644–654.

[13] C. Karlof, D. Wagner, Secure routing in sensor networks: attacks and countermeasures, First IEEE International Workshop on Sensor Network Protocols and Applications (2003).

[14] A.D. Wood, J.A. Stankovic, Denial of service in sensor networks, Computer 35 (10) (2002) 54–62.

[15] C. Karlof, N. Sastry, D. Wagner, TinySec: a link layer security architecture for wireless sensor networks, Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004) (2004).

[16] F. Ye, H. Luo, S. Lu, L. Zhang, Statistical en-route detection and filtering of injected false data in sensor networks, Proceedings of IEEE INFOCOM (2004).

[17] D.W. Carman, B.J. Matt, G.H. Cirincione, Energy-efficient and low-latency key management for sensor networks, Proceedings of 23rd Army Science Conference (2002).

[18] M. Chen, W. Cui, V. Wen, A. Woo, Security and Deployment Issues in a Sensor Network, Ninja Project, A Scalable Internet ServicesArchitecture, Berkeley.

[19] W. Zhang, G. Cao, Group rekeying for filtering false data in sensor networks: a predistribution and local collaboration-based approach, Proceedings of INFOCOM (2005).

[20] D. Liu, P. Ning, K. Sun, Efficient self-healing group key distribution with revocation capability, Proceedings of the 10th ACM conference on Computer and communication security (2003).

[21] J. Lee, D.R. Stinson, Deterministic key pre-distribution schemes for distributed sensor networks, To appear in Lecture Notes in Computer Science (SAC 2004 Proceedings) (2004).

[22] A. Leon-Garcia, Probability and Random Processes for Electrical Engineering, 2nd ed., Addison-Wesley Publishing Company Inc., Reading, MA, 1994.

[23] S. Shakkottai, R. Srikant, N. Shroff, Unreliable sensor grids: coverage, connectivity and diameter, in: Proceedings of the IEEE INFOCOM, 2003, pp. 1073–1083.

[24] X. Du, Y. Xiao, M. Guizani, H.H. Chen, An Effective Key Management Scheme for Heterogeneous Sensor Networks, Ad Hoc Networks, Elsevier, vol. 5, issue 1, January 2007, pp. 24–34.

[25] X. Du, M. Guizani, Y. Xiao, S. Ci, H.H. Chen, A routing-driven elliptic curve cryptography based key management scheme for heterogeneous sensor networks, in: IEEE Transactions on Wireless Communications, accepted for publication (to appear).

[26] D. Malan, M. Welsh, M.D. Smith, A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography, in: Proceedings of 1st IEEE International Conference Communications and Networks (SECON), Santa Clara, CA, October 2004.

[27] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, Comparing elliptic curve cryptography and RSA on 8-bit CPUs, in: Proceedings of the 6th International Workshop on Cryptographic Hardware and Embedded Systems, Boston, Massachusetts, August 2004.

[28] A.S. Wander, N. Gura, H. Eberle et al., Energy analysis of public-key cryptography for wireless sensor networks, in: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM), 2005.

[29] P. Traynor, H. Choi, G. Cao, S. Zhu, T. Porta, Establishing pair-wise keys in heterogeneous sensor networks, in: Proceedings of IEEE INFOCOM 06.

[30] M. Eltoweissy, M. Moharrum, R. Mukkamala, Dynamic key management in sensor networks, IEEE Communications Magazine 44 (4) (2006) 122–130.

[31] P. Ning, R. Li, D. Liu, establishing pairwise keys in distributed sensor networks, ACM Transactions on Information and System Security 8 (1) (2005) 41–77.

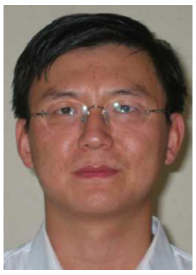
[32] F. Anjum, Location dependent key management using random key-predistribution in sensor networks, in: Proceedings of WiSe'06.

[33] M.F. Younis, K. Ghumman, M. Eltoweissy, Location-aware combinatorial key management scheme for clustered sensor networks, IEEE Transactions on Parallel and Distributed Systems 17 (8) (2006) 865–882.

[34] W. Du, J. Deng, Y. Han, P.K. Varshney, A pairwise key predistribution scheme for wireless sensor networks, in: Proceedings of CCS'03.

Yang Xiao worked at Micro Linear as an MAC (Medium Access Control) architect involving the IEEE 802.11 standard enhancement work before he joined Department of Computer Science at The University of Memphis in 2002. Dr. Xiao is currently with Department of Computer Science at The University of Alabama. He was a voting member of IEEE 802.11 Working Group from 2001 to 2004. He is an IEEE Senior Member. He is a member of American Telemedicine Association. He currently serves as Editor-in-Chief for *International Journal of Security and Networks (IJSN)*, *International Journal of Sensor Networks (IJSNet)*, and *International Journal of Telemedicine and Applications (IJTA)*. He serves as a referee/reviewer for many funding agencies, as well as a panelist for NSF and a member of Canada Foundation for Innovation (CFI)'s Telecommunications expert committee. He serves as TPC for more than 90 conferences such as INFOCOM, ICDCS, ICC, GLOBECOM, WCNC, etc. His research areas are wireless networks, mobile computing, network security, and telemedicine. He has published more than 200 papers in major journals (more than 50 in various IEEE Journals/magazines), refereed conference proceedings, book chapters related to these research areas. Dr. Xiao's research has been supported by NSF.

Venkata Rayi did his master's program in the Computer Sciences Department at The University of Memphis, Memphis, TN 38152 USA (e-mail: Krishna.rayi@gmail.com). He is right now working as Software Engineer for a software firm called XINOPT in the state of MD.



Bo Sun (bsun@cs.lamar.edu) received his Ph.D. degree in Computer Science from Texas A&M University, College Station, U.S.A., in 2004. He is now an assistant professor in the Department of Computer Science at Lamar University, U.S.A. His research interests include the security issues (intrusion detection in particular) of Wireless Ad Hoc Networks, Wireless Sensor Networks, Cellular Mobile Networks, and other communications systems. His research is supported by 2006 Texas Advanced Research Program (ARP) and NSF DUE-0633445.

Xiaojiang (James) Du is an Assistant Professor in the Department of Computer Science, North Dakota State University. Dr. Du received his B.E. degree from Tsinghua University, Beijing, China in 1996, and his M.S. and Ph.D. degrees from University of Maryland, College Park in 2002 and 2003, respectively, all in Electrical Engineering. His research interests are heterogeneous wireless sensor networks, security, wireless networks, computer networks, network and systems management, and controls. Dr. Du is an Associate Editor of *Wireless Communication and Mobile Computing* (Wiley), and *International Journal of Sensor Networks* (InderScience). He is (was) the Chair of Computer and Network Security Symposium of ACM International Wireless Communication and Mobile Computing Conference 2007 (2006). He is (was) a TPC member for several major IEEE conferences such as INFOCOM, ICC, GLOBECOM, WCNC, IM, NOMS, BroadNet and IPCCC.

Fei Hu received the BS and MS degrees from Shanghai Tongji University (China) in 1993 and 1996, respectively. He received the Ph.D. degree from the Department of Electrical and Computer Engineering at Clarkson University in 2002. His Ph.D. research was on high-performance transmission issues in wireless networks. He is currently an assistant professor in the Computer Engineering Department at RIT, New York. He served as a senior networking engineer at the Shanghai Networking Lab and Shanghai Lucent Inc. from 1996 to 1999, where he worked on several large projects on high-performance networks. Dr. Hu is a full Sigmaxi member, a member of the IEEE, and an IEEE chapter officer. His research interests are in ad hoc sensor networks, 3G wireless and mobile networks, and network security. His research has been supported by NSF, Cisco, Lockheed Martin, Sprint, and so on.

Michael Galloway received the A.A.S. degree in Electronics Technology from Central Alabama Community College, Alexander City, AL in 2001. He received the B.W.E. degree in Wireless Software Engineering from Auburn University, Auburn, AL in 2005. He is currently enrolled as a Ph.D. student in the Department of Computer Science at the University of Alabama. His current research interests are wireless networks, security in wireless networks, cognitive radios, routing protocols, and the OSI model. He is a student member of IEEE and ACM. He is a member of the SIGMOBILE interest group within ACM.