Winter 2006

Group Key Management in Wireless Ad-Hoc and Sensor Networks

Mohammed A. Moharrum
Old Dominion University

GROUP KEY MANAGEMENT IN WIRELESS AD-HOC AND SENSOR NETWORKS

by

Mohammed A. Moharrum
M.Sc. August 2000, Alexandria University, Egypt
B.Sc. June 1997, Alexandria University, Egypt

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

OLD DOMINION UNIVERSITY

December 2006

Approved by:

Ravi Mukkamala(Director)

Mohamed Eltoweissy
(Member)

Stephan Olariu (Member)

Hussein Abdel-Wahab
(Member)

Min Song (Member)

ABSTRACT

GROUP KEY MANAGEMENT IN WIRELESS AD-HOC AND SENSOR NETWORKS

Mohammed A. Moharrum
Old Dominion University, 2006
Director: Dr. Ravi Mukkamala

A growing number of secure group applications in both civilian and military domains is being deployed in WAHNs. A Wireless Ad-hoc Network (WAHN) is a collection of autonomous nodes or terminals that communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner. A Mobile Ad-hoc Network (MANET) is a special type of WAHN with mobile users. MANET nodes have limited communication, computational capabilities, and power. Wireless Sensor Networks (WSNs) are sensor networks with massive numbers of small, inexpensive devices pervasive throughout electrical and mechanical systems and ubiquitous throughout the environment that monitor and control most aspects of our physical world.

In a WAHNs and WSNs with un-trusted nodes, nodes may falsify information, collude to disclose system keys, or even passively refuse to collaborate. Moreover, mobile adversaries might invade more than one node and try to reveal all system secret keys. Due to these special characteristics, key management is essential in securing such networks. Current protocols for secure group communications used in fixed networks tend to be inappropriate. The main objective of this research is to propose, design and evaluate a suitable key management approach for secure group communications to support WAHNs and WSNs applications.

Key management is usually divided into key analysis, key assignment, key generation and key distribution. In this thesis, we tried to introduce key management schemes to provide secure group communications in both WAHNs and WSNs.

Starting with WAHNs, we developed a key management scheme. A novel architecture for secure group communications was proposed. Our proposed scheme handles key dis

tribution through Combinatorial Key Distribution Scheme (CKDS). We followed with key generation using Threshold-based Key Generation in WAHNs (TKGS). For key assignment, we proposed Combinatorial Key Assignment Scheme (CKAS), which assigns closer key strings to co-located nodes. We claim that our architecture can readily be populated with components to support objectives such as fault tolerance, full-distribution and scalability to mitigate WAHNs constraints. In our architecture, group management is integrated with multicast at the application layer.

For key management in WSNs, we started with DCK, a modified scheme suitable for WSNs. In summary, the DCK achieves the following: 1) cluster leader nodes carry the major part of the key management overhead; (2) DCK consumes less than 50% of the energy consumed by SHELL in key management; (3) localizing key refreshment and handling node capture enhances the security by minimizing the amount of information known by each node about other portions of the network; and (4) since DCK does not involve the use of other clusters to maintain local cluster data, it scales better from a storage point of view with the network size represented by the number of clusters.

We went further and proposed the use of key polynomials with DCK to enhance the resilience of multiple node capturing. Comparing our schemes to static and dynamic key management, our scheme was found to enhance network resilience at a smaller polynomial degree t and accordingly with less storage per node.

This dissertation is dedicated to the sole of my late father; he was hoping to see this day. God bless him.

ACKNOWLEDGMENTS

Thanks to God who guided me to this, and I would never reach this without his guidance.

This dissertation would not have been successfully completed without the contributions of a number of people. My deepest gratitude and appreciation are due to my advisors, Dr. Mukkamala and Dr. Eltoweissy, who helped a lot in finishing this work.

In addition, I would like to extend my thanks to Dr. Abdel-wahab and Dr. Olariu, who both helped a lot in getting this work done.

My motivating supportive family has always been a source of encouragement during my study. My wife and my daughter inspired me to achieve my goal. My parents and brother always supported me during this journey, although that meant we would be separated thousands of miles for extended periods of time. My utmost thanks for my entire family for their everlasting support and encouragement.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A wireless ad-hoc network (WAHN) is a collection of autonomous nodes or terminals that communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner. Mobile Ad-hoc Networks (WAHNs) are special type of WAHNs with mobile users. The unique characteristics of WAHNs[1], including limited power, communication and computation capabilities, have introduced many research challenges. One major characteristic is the absence of any type of fixed communication infrastructure support as well as the very limited resources of nodes. Researchers trying to implement well-defined services from other networking domains cannot just automatically transfer such schemes. Due to these characteristics, WAHNs or WAHNs call for distributed collaborative schemes to perform different network functionalities. On the other hand, if those nodes are smaller in size and computation capabilities, they are known as Wireless Sensor Networks (WSNs).

In an ad-hoc network, secure communication is essential to different types of applications. The simplest case is secure communications between two nodes (*1-to-1*). Several researchers have proposed solutions for this type of applications [132]. Other applications, such as mobile advertisement or secure broadcast of commands in military tactical operations, might require a single sender to deliver messages to multiple receivers (*1-to-M*). Furthermore, applications such as mobile auctioning or collaboration of military units in tactical operations, and distributed virtual environments (DVEs), require secure group communications (*M-to-M*). In addition to two types of group applications, other non-group applications, such as fusing sensor readings to report a phenomenon, secure routing and exchange of network status information, might also require secure team collaboration. Such WAHNs applications might take good advantage of an underlying secure group communication service.

The formatting in this thesis follows the IEEE Transaction in Software Engineering guidelines.

Providing efficient and secure group communication services in WAHNs environments, however, encounters many challenges. Typically, nodes in WAHNs have limited storage and communications resources. In addition to these bottlenecks, the lack of any networking infrastructure, the transient nature of connections, and the overhead introduced by implementing security, further exacerbate the problem [5]. This thesis addresses some of the challenges of providing secure and efficient group communication services in wireless ad-hoc network environments with medium (hundreds) to large (thousands) number of un-trusted nodes.

In military tactical operations, fast and secure communications need to be established between different units deployed in an unknown hostile territory. Those units may include different type of vehicles or individual military personnel as well as small aircraft. Such units may move along unplanned paths and routes due to unexpected circumstances. Moreover, no networking infrastructure or fixed support of any can be assumed. Thus, those units need to communicate securely and in an autonomous fashion. Since ad-hoc networks technology was initially developed with military applications in mind [84], the above situation is a typical military application of MANETs, with secure group communication requirements In addition to the above, it is natural to assume that some units might be compromised, captured or failed. Such nodes should not affect the communications and collaboration of other nodes. **Fig. 1** depicts such an application. The operation involves multiple vehicles and three aircrafts. Not all units can reach each other directly. However, a path can be provided through hop-by-hop routing. Some enemy aircrafts might be listening to the communications. They should not be able to interpret. At the same time, one vehicle, shown in the picture, was captured and compromised. This vehicle is trying to mislead other units. Another vehicle lost power source and was abandoned. A robust secure group communications protocol for MANETs is required to survive such hostile circumstances.

Current M-Commerce applications depend on cellular communications infrastructure support. Although cellular networks cover global areas, most M-commerce applications such as mobile advertising or mobile auctioning are inherently local. In the future, local

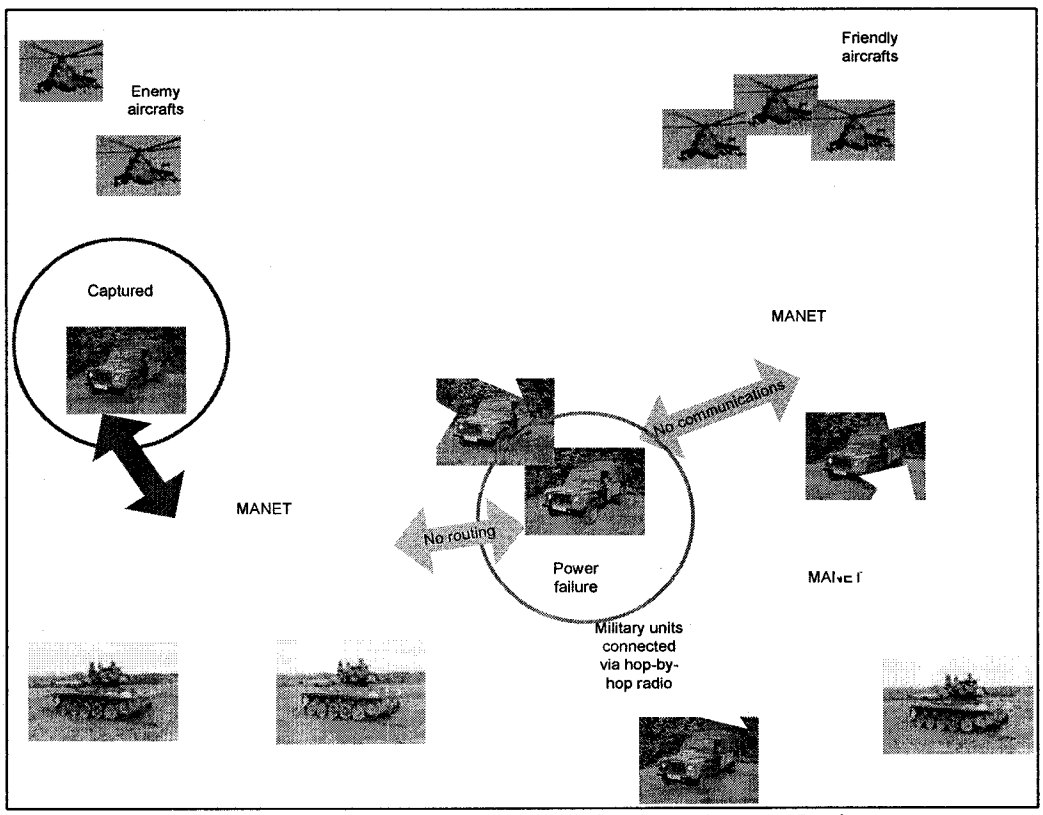

Fig. 1. A Small Military tactical Operation in Unknown Territory.

cellular networks might be replaced with set of autonomous self-organizing MANET-based devices. From an economical point of view, efficient MANET devices provide a

cost effective alternative for localized mobile applications. One reason is that a third party is not involved and does not charge transaction fees. Another reason is that no expensive mobile support stations are needed. This alternative support should be integrated with the original fixed support in a seamless way that is transparent to users. Consider applying this alternative to an existing M-Commerce application such as a mobile auctioning system. In mobile auctioning, both parties, sellers and bidders, can be mobile. They use their cellular phones for performing transactions such bidding or advertising. A cellular network provider facilitates such transaction through its cellular coverage. Replacing this with MANET-based system, we get the situation in Fig. 2, where sellers and bidders are in mobile vehicles on a highway. Some other fixed stations might exist on the highway. However, those stations are not part of the system. For instance, some individuals living close to the highway may casually subscribe to the service if they are interested. In such a case, a seller auctions his item using a MANET-device (such as a laptop) on his vehicle. Mobile subscribers may receive such information and can even place bids. Other mobile vehicles may not receive, or be able to interpret, such messages. However, they may voluntarily provide routing points through their vehicles as part of the ad-hoc networking infrastructure. As some vehicles move away, they might not be interested in the service anymore. At the same time, other vehicles may approach the area and join the service. This should not interfere with the service to other subscribers. Since moving vehicles represent a consciously dynamic routing topology, this is a typical MANET application. Since security is essential to M-Commerce transactions, the above application calls for secure group communication service in a mobile ad-hoc network.

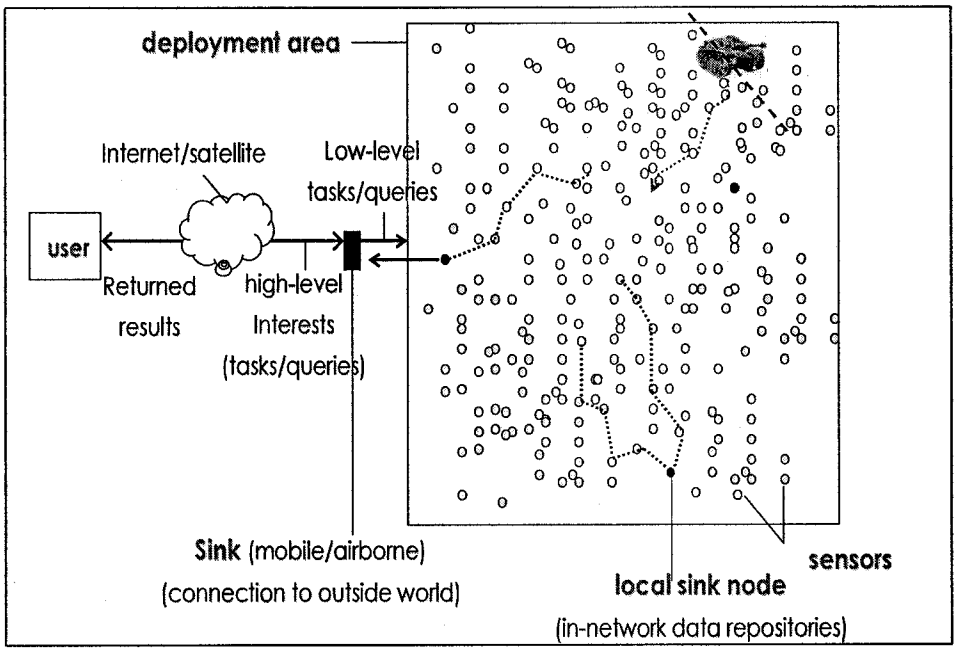

Fig. 2. Structure of a Typical Sensor Networks.

Fig. 2 shows the structure of a sensor network.

The main difference between WAHNs and WSNs are

- Number of nodes: in WSNs they are orders of magnitude larger than WAHNs,
- Reliability: Sensors are more prone to failure,
- Topology: more dynamic in WSNs due to sleep-awake cycle,
- Communication: broadband in WSNs while a point-to-point in WAHNs,
- Resources: the power, computation and communications are lower in WSNs than WAHNs,
- Sink: different from WAHNs, a long-range radio connecting WSNs to the outside world.

However, in both networks, secure communication is essential to different type of applications. The simplest case is secure communications between two nodes (*1-to-1*). Several researchers have proposed solutions for this type of applications [132]. Other applications, such as mobile advertisement or secure broadcast of commands in military tactical operations, might require a single sender to deliver messages to multiple receivers (*1-to-M*). .

1.1. OVERVIEW

Providing efficient and secure group communication services in WAHNs and WSNs environments, however, encounters many challenges. Typically, nodes in WAHNs have limited storage. The computational and communications resources are under limited in WSNs than in WAHNs. In addition to these bottlenecks, the lack of any networking infrastructure, the transient nature of connections, and the overhead introduced by implementing security, further exacerbate the problem. [5]. This thesis addresses some of the challenges of providing secure and efficient group communication services in wireless ad-hoc network environments with medium (hundreds) to large (thousands) number of un-trusted nodes. It also addresses WSNs with even a larger size.

A growing number of secure group applications are expected to be delivered in WAHNs and WSNs environments for their applicability in civilian and military domains. The two major requirements for supporting such applications are group communications and secure group management. Currently, group communications is supported through network level multicast routing protocols (e.g. MAODV, OMRP,.., etc). Such protocols provide basic session management in addition to multicast data delivery. However, a considerable amount of group state information is stored at individual nodes. Moreover, such protocols do not provide any type of group security.

In wire-line networks, secure group management have been investigated thoroughly and many good protocols have been proposed. Since such protocols do not take in consideration the special characteristics of WAHNs and WSNs. They are not automatically applicable to WAHN group applications. At the same time, current group applications based on network-level multicast support tend to redundantly perform some group management tasks at both application and network layer.

1.2. CONTRIBUTIONS

In this thesis, we propose a new framework for secure group applications in WAHNs and WSNs. In our model for WAHNs, we assume an internal threat from autonomous un-trusted nodes that might falsify information, collude to reveal secret keys, or simply choose not to collaborate. Our external threat includes mobile adversaries that can compromise a number of nodes as well. We use Application-Level Multicast (ALM) as the means of group communications rather than network-level multicast routing protocol. We propose a new overlay architecture that provides an abstraction for the underlying network dynamic topology.

Among the several modules of the proposed architecture, group management is especially sensitive to the characteristics of the underlying infrastructure. While several key management protocols such as Group Key Management Protocol (GKMP) [49]), Logical Key Hierarchy (LKH) [22], and One-way Function Chain Tree (OFCT) [12] have been proposed in literature, we find that they are not directly applicable to WAHN environments. Other researchers adopted some key agreement protocols from distributed peer systems to WAHNs, e.g. [119]. However, most of these protocols depend on variations of Diffie-Hellman protocol, which might require both reliable communications as well as significant computations. Here, we propose and integrate schemes for different group management components (e.g., key generation, key assignment, and key distribution). Our proposed scheme handles key distribution through Combinatorial Key Distribution Scheme (CKDS). We followed with key generation using Threshold-based Key Generation in WAHNs (TKGS). For key assignment, we proposed Combinatorial Key Assignment Scheme (CKAS), witch assigns closer key strings to co-located nodes. We show that our architecture can readily be populated with components to support objectives such as fault tolerance, full-distribution and scalability to mitigate WAHNs constraints. In our architecture, group management is integrated with multicast at the application layer. We also show how each component supports security, fault–tolerance, and efficiency.

Considering WSNs, we propose a similar solution through an efficient EBS-base key management scheme. Our proposed scheme was shown to overcome several problems in other schemes such as (1) cluster leader nodes carry the major part of the key

management overhead; (2) Consuming less than 50% of the energy consumed by other schemes in key management; (3) localizing key refreshment and handling node capture enhances the security by minimizing the amount of information known by each node about other portions of the network; (4) Using key polynomials enhance network resilience at a smaller polynomial degree t and accordingly with less storage per node.

1.3. OUTLINE

The remaining part of this thesis is organized as follows. In Chapter 2, we describe the necessary background material, including group communications in WAHNs and WSNs, Key management schemes for both, and the need for improvement. In Chapter 3, we describe the proposed solutions for key management in WAHNs. In Chapter 4, we show the simulation results comparing our solution to existing ones. In Chapter 5, we describe our proposed key management scheme for WSNs. In Chapter 6, we show the simulation results comparing our scheme to existing ones in WSNs. In Chapter 7, we conclude by summarizing our contribution as well as presenting ideas for future research.

CHAPTER II

BACK GROUND AND RELATED WORK

A Wireless Ad-Hoc Network (WAHN) is a collection of autonomous nodes or terminals that communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner. Mobile Ad-hoc Networks (WAHNs) are special type of WAHN with mobile users [89]. The unique characteristics of WAHNs, such as limited power, communication and computation capabilities, have introduced many research challenges. One major challenge is preserving the efficient and secure communications in the absence of any type of fixed communication infrastructure support as well as the very limited resources of nodes. Researchers trying to implement well-defined services from other networking domains cannot just automatically transfer such schemes.

Wireless Sensor Networks (WSNs) usually use micro-sensor technology with no fabrication-time identity as well as low-power signal processing and computations to reduce the size as well as extending the life with bob-renewable energy source.

Due to the critical applications of both types of networks in both military and civilian arenas, security is a major concern [15]. Confidentiality, authenticity, availability and integrity are the major security objectives. WAHNs are subject to different types of attacks from a variety of attackers that target one or more of these security goals. Attacks may be external varying from traffic jamming to compromising nodes, or internal such as collusion.

Secure communication among WAHNs and WSNs nodes is an essential requirement mandated by many applications and lack of which might highly restrict the applicability of such networks. There are several solutions to the secure communication problem depending on the communication model. Some solutions assume a peer-to-peer communication model and thus construct (or pre-distribute) pair-wise keys. Other solutions assume a group communication model in which a single source multicasts message to numerous destinations. In such a case, group keys are established.

The main objectives of this chapter are to discuss some security aspects in WAHNs and WSNs and some existing solutions and protocols for secure group communication. This will lead to the need of new key management solutions for both types of networks which is the main contribution of this thesis.

The structure of this chapter is organized as follows. Section 2.1 discusses architectures, components and characteristics of both WAHNs and WSNs. Section 2.2 discusses security objectives, threats and attacks in WAHNs and WSN's. Section 2.3 discusses secure communication schemes and key management protocols in WAHNs and WSNs. In section 2.4, we briefly describe existing group key management protocols in WAHNs. In section 2.5, we briefly describe existing group key management protocols in WSNs. In section 2.6, we briefly describe Exclusion-Basis Systems (EBS) as the theoretical foundation of our proposed solutions. Finally, in section 2.7 we summarize the existing solutions and establish the need for new solutions.

2.1. WIRELESS AD-HOC NETWORKS AND WIRELESS SENSOR NETWORKS ARCHITECTURES

WAHNs and MANETs are the new generation of wireless devices. They are receiving much attention from academia, industry, and government. Although they have many applications in civilian and military arenas, certain specific characteristics appear to be very challenging in designing solutions to such networks [5]. In the first sub-section, we briefly introduce the reader to Wireless Ad-hoc Networks architecture. In the second sub-section, we discuss the Wireless Sensor Networks architecture.

2.1.1. THE WAHN ARCHITECTURE

2.1.1.1. WAHN COMPONENTS (NODES)

A wireless ad-hoc network is a collection of autonomous nodes or terminals. Each node in a wireless ad-hoc network functions as both a host and a router, and the control of the network is distributed among the nodes [5]. The number of nodes in such a network varies between few tens in MANETs and up to millions in sensor networks.

In an ad-hoc network, nodes may move as a result, the signal strength between any two nodes may change. In addition, the environment has a greater effect on the signal strength. Electromagnetic interference can reduce signal strength. When signal strength is reduced, protocols often reduce their bandwidth requirements. Increased traffic on the radio frequency can also reduce bandwidth.

Users may come and go at any time resulting in routing problems. This may be due to energy levels, as in sensor networks, or just the nature of the users, as in the ad-hoc Internet network connection. Another problem is that nodes may lose enough signal strength to another node that they effectively become unavailable.

There is no mechanism to physically secure network connections. For example, a secure wired network might have computers secured inside a locked building, connected by fiber optic cable, encased in pressurized conduit. If the pressure drops in the conduit an alarm is sounded. On the other hand, in an ad-hoc wireless network, even a novice hacker with the right brand of WI-FI card can gain access to the network communication data.

Nodes may be under power limitations as they may be powered by battery or other low power sources. In addition, nodes may only be available at certain times. This could be due to power conservation [106], or because the node must operate in "stealth mode" as in a military situation.

2.1.1.2. WAHN CONNECTORS (COMMUNICATION)

Nodes communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner. Since WAHN nodes communicate over wireless links, they have to contend with the effects of radio communication such as noise, fading, and interference. In addition, the links typically have less bandwidth than in a wired network. The network topology, in general, is dynamic, because the connectivity among the nodes may vary with time due to node departures, new node arrivals, and the possibility of having mobile nodes. Hence, there is a need for efficient routing protocols to allow the nodes to communicate over multi-hop paths consisting of possibly several links in a way that does not use any more of the network "resources" than necessary.

Nodes in an ad-hoc network are required to provide routing services. Routing is usually provided at the network level. A number of routing protocols have been proposed such as AODV [90] and DSR [57] and many others. Routing without security can cause major problems. Multicast routing may be provided at the network level as well. Example protocols include MADOV [99] and ODMRP [64]. Application-level multicast achieves more scalability and implements more sophisticated group logic at the application layer. Examples include AMRoute protocol [117], and PAST-DM [44].

An ad-hoc network may be a self-contained network, act as a stub [22], or be used as a connection between two networks. A self-contained network is the simplest because it can run custom routing protocols.

In some ad-hoc networks (e.g. MANETs), nodes are free to move arbitrarily at different speeds. In other networks, the random ad-hoc interaction among nodes (e.g., smart sensor networks) produces a dynamic networks topology [5].

2.1.1.3. WAHN CONFIGURATIONS

Based on the node mobility, wireless ad-hoc networks may be classified to be either fixed or mobile. Mobile Ad-hoc Networks (MANETs) is an example of mobile WAHN. A MANET is an autonomous collection of mobile users that communicate over bandwidth-constrained wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. An example of fixed ad-hoc networks is Smart Sensor Networks (SSN). A smart sensor network is a large number of sensor nodes deployed randomly and communicate via short-range radio transmission [5]. Nodes spend most of their time in a sleep mode to save energy and wake up randomly to interact with each other in an ad-hoc fashion.

Based on membership restrictions, WAHNs may also be classified into open or closed classes. In an open network, anyone can join. A new user would contact nearby users and set up communication. The node then may be asked to route traffic from one node to another, and can access other nodes through its neighbors. An example would be an Armature Packet Radio Network (APRN). Anyone with a ham radio license can join the network. Obviously, the ability for anyone to join a network is itself a serious security concern.

In a closed network, only valid nodes can join. A node is either pre-configured with keys or has some special information that only that ad-hoc group knows. The join is then done in a very similar manner to open networks with the exception that the initialization may take extra steps. The security concerns for closed networks are much like that of standard WI-FI networks. The general idea is simply to keep intruders out.

The main distinction between WAHNs and other infrastructured wireless communication systems is the lack of any centralized control or fixed support stations found in cellular wireless networks (e.g., MSS, MSC, HLR, VLR, etc) [48], Although most research work assumes pure WAHNs (e.g. AODV [90] and DSR [57] and many others), hybrid configuration is also viable. Even within pure WAHNs, nodes might not be of the same capabilities. For example, in smart sensor networks, a sensor node normally is assumed to have up to 4MHz processor and 8KB memory [92]. At the same time, the network deploys one or more special nodes that, with larger capabilities, can communicate with the base station.

Another example of hybrid networks is integrating cellular networks with MANETs to improve QoS such as in Cellular-aided mobile Ad-hoc Network (CAMA) [48]. In this architecture, ad-hoc nodes register at the cellular service provider for authentication. Network management tasks are performed through cellular servers while data exchange is done through the ad-hoc network.

No assumption can be made about node collaboration. Nodes might refuse to get involved in any collaborative group management function either deliberately or unwillingly due to limited resources. WAHNs nodes are generally more prone to physical security threats than fixed cable nets. These threats include, but not limited to, eavesdropping, spoofing, and denial-of-service-like attacks [51].

Due to the above characteristics, regular network protocols employed in fixed wire line networks are not transparently transferable to WAHNs. WAHNs' characteristics mandate careful design of routing protocols as well as mobility management algorithms to increase reliability and availability. All algorithms and protocols should focus on the limited capabilities of WAHN nodes to meet the above challenges. Secure group communications, which is the focus of this work, is no exception.

2.1.2. WIRELESS SENSOR NETWORK ARCHITECTURE

A sensor network is usually assumed to have a *base station* and a large number of identical *sensor nodes* [31] [32] [33]. No pre-determined configuration is assumed. Upon deployment, the base station and the sensor nodes work together to dynamically establish a virtual configuration. Flat (or single level) sensor networks lack scalability and are usually limited by the transmission range of the base station [14]. A hierarchical (multilevel) sensor network consists of a *base station*, several *data aggregation nodes*, and a large number of *sensor nodes*. Such architecture was shown to provide scalability, notable energy efficiency, and security benefits [122] and [14]. Our system model adopts the latter model and distributes key management tasks among the network levels.

For the purpose of key management, we consider a two-level (three-tier) sensor network consisting of a topmost *base-station (BS),* under which, *sensor nodes (SN)* are organized in clusters. Each cluster is lead by a node known as *cluster leader (CL)*. CLs comprise the second highest tier while sensor nodes are at the bottom tier. Clusters can be formed based on various aspects such as capabilities, location, communication range, etc. [46]. Cluster leaders are able to communicate with the base station (also known as command node). In addition, we assume that in each cluster, the cluster leader is capable of reaching all other nodes within the cluster through broadcast, and hence it plays the role of a Key Distribution Node (KDN). In this thesis, we assume that both sensor nodes and cluster leaders are stationary and that each node is aware of its physical location and communication range. In contrary to SHELL [122], DCK cluster leaders are different from data gateways. A cluster leader is considered a keying gateway that performs certain key management function, yet, cluster leaders are not necessarily data gateways that perform data aggregation and processing functions.

Each tier of the network possesses different architectural capabilities. The base station is assumed to have no computational and storage limitations. However, the communication channel between the base station and the cluster leaders is assumed to be restricted. Since the base station might not be collocated with the other nodes in the deployment field, communications between nodes and the base station might take place either though slow satellite links and/or multi-hop packet transmissions, which might increase the probability of packet loss. This motivates the minimization of the base station

involvement in the regular network operations. A sample sensor network is depicted in Fig. 3.

Similar to SHELL [122] and SECK [14], DCK assumes cluster leaders to enjoy better capabilities than regular sensor nodes. The sensor nodes are assumed to be constrained in energy and processing resources, and cannot perform complex computations or store large amounts of data. The MICA Mote, which includes 4K RAM and an 8-bit 4MHz processor, and runs with 2 AA batteries, is an example for the capabilities of a typical sensor node.

2.2. SECURITY OBJECTIVES IN WAHNS AND WSNS

The basic method for securing communications is cryptography. Confidentiality, integrity and availability are the main goals of securing communications in WAHNs [51]. Although encryption/decryption is meant to provide data confidentiality, cryptographic techniques provide integrity and may even contribute to availability. We start by discussing briefly the security objectives that applies to both WAHNs and WSNs. Then, we discuss different threat models and attacks on WAHNs. Then, we discuss different threat models and attacks on Security Objectives in WSNs.

2.2.1. SECURITY OBJECTIVES

The major objectives of securing communications in WAHNs and WSNs are confidentiality, integrity, authenticity, and availability. In this sub-section, we introduce the need for these objectives in applications

2.2.1.1. CONFIDENTIALITY

Confidentiality ensures that certain information is never disclosed to unauthorized entities. Data encryption (and decryption) with either symmetric or asymmetric keys clearly provides data confidentiality (e.g., against eavesdropping). In the context of pair-wise communications, confidentiality can be defined as the property of hiding the communicated data from anyone other than the two communicating parties. In the context of group communications, where more than two parties are involved, confidentiality (group commutations people prefer the term secrecy [116]) is more

complicated and time dependent. *Current secrecy* refers to the property that any party other than the *current* communication group members is unable to obtain the communicated data. *Forward secrecy* refers to the property that previous members of the communicating group should be unable to obtain data communicated after they have left the group. *Backward Confidentiality* refers to the property that new members of the communicating group should be unable to obtain data communicated before they have joined the group. In WAHNs, hop-by-hop radio is used as the communication channel [5]. In some applications, such as military tactical operations or other civilian applications described above, other parties might interfere with this communications. Thus confidentiality is an essential security goal in WAHNs applications.

2.2.1.2. INTEGRITY

Integrity is the property that guarantees that a message being transferred is never corrupted. A message could be corrupted because of benign failures, such as radio propagation impairments, or because of malicious attacks on the network [130]. Message Authentication Codes (MAC) is a cryptographic technique used for assuring data integrity. Deng *et al* [23] used such technique to achieve security and integrity of routing information in MANETs.

2.2.1.3. AVAILABILITY

Availability can be defined as the property of maintaining the network (or network services) up and running under hostile conditions (such as denial of service attacks) [130]. A denial of service attack could be launched at any layer of an ad-hoc network. On the physical and media access control layers, an adversary could employ jamming to interfere with communication on physical channels. On the network layer, an adversary could disrupt the routing protocol and disconnect the network. On the higher layers, an adversary could bring down high-level services. One such target is the key management service, an essential service for any security framework [48].

2.2.1.4. AUTHENTICATION

Source authentication is the property that enables a receiver to verify that the received data has originated from the claimed source and has not been compromised on the route [48]. Authentication in WAHNs may be achieved using either asymmetric key operations through public-key infrastructure (PKI), or symmetric key operations.

Asymmetric key cryptography is the main authentication technique in modern computers. Public Key Infrastructure (PKI) is used everywhere on the Internet to authenticate users and service providers different transactions [115]. The major problems with PKI in WAHNs are the need for a centralized Certificate Authority (CA) its availability, and Certificate Revocation Lists (CRLs) [1]. Some decentralized *Public Key Infrastructure* (PKI) solutions have been proposed in WAHNs. Threshold protocols have been used quite successfully in developing PKI-based security scheme with a threshold-based (Certification Authority) CA such as COCA [130]. In MANETs, a similar technique was proposed in MOCA [126]. A performance enhancement by using cashing was proposed in CACMAN [1]. Although PKI can be deployed in WAHNs with certain capabilities, it might not be suitable for other ad-hoc networks with more restricted capabilities (e.g., sensor networks). The main reason, in addition to the need for an efficient CA, is the computational complexity and storage overhead of asymmetric key operations. Brown *et al* [10] have reported that a 512-bit RSA signature generation takes 2.6 seconds on a RIM Pager and a Palm Pilot. Perrig *et al* [92] report that a current generation sensor node has just 4500 bytes for security and application needs.

Symmetric key operations are much more affordable for WAHNs due to their lower overhead and storage requirements. Existing symmetric key authentication schemes in wired networks, such as Kerberos [61] and Otway-Rees [88], cannot fit in WAHNs due to their computational overhead. Reserchers proposed several lighter weight authentication schemes for WAHNs. Examples include TESLA [93], LEAP [134] and μTESLA [92].

2.2.2. A THREAT MODEL FOR WHANS

Threats can be characterized based on as to the area of attack. Common domains of security compromise include physical, personnel, hardware, software, and procedural

[37]. All of these areas do not apply directly to ad-hoc networks, but nevertheless they are valuable security areas for the entire life cycle of networks architectures. Attackers can be either internal or external. In the following, we briefly discuss relevant types of attacks and attackers to our WAHN.

2.2.2.1. THREATS

Like all other networks, WAHNs are subject to attacks of different types. According to Farhmand [37], attacks may be physical, personnel, hardware, software, or procedural.

a. Physical: Physical penetration refers to the physical means of gaining access to a restricted area. Areas can include buildings, computer labs, or other sensitive domains. Physical attacks are not common themes in the deployment of ad-hoc network security.

b. Personnel (collusion): An individual who subverts personnel authorization, and gains unauthorized access or privilege are said to have penetrated area of personnel. Ad-hoc nodes, who are authorized nodes, and then abuse that status by colluding to re-configure the network, compromise secret keys, or route packets to an unauthorized node has violated are area of personnel.

c. Hardware: A hardware attack can be mounted in order to subvert or a deny service to a system. Ad-hoc networks can be vulnerable to this in the case of human sabotage. A device that is not tamper-resistant might be vulnerable to this type of attack. In addition, building custom hardware to circumvent a network's security measures could be considered a hardware attack.

d. Software: Techniques, which compromise the integrity of system software, application programs, or utility routines, are attacks against the software.

e. Procedural: A procedural penetration is where an authorized or unauthorized intruder node can gain access to the system. This threat arises when nodes leave or join networks, and the confidentiality of the private keys must be protected.

In addition to the above network security attacks, some reports on ad-hoc security have the additional measure of privacy. Furthermore, attacks are qualified based on whether they are passive or aggressive and in terms of the network layer to which it is aimed.

Finally, attacks can be qualified according to the network layer to which they are aimed.

Application Layer: An attack that, for instance, injects false information to undermine the integrity of an application is referred to as an Application Layer attack.

Network Layer: A node with access to the routing mechanisms can degrade the network. This also includes a node that acts selfishly, not working cooperatively with the network.

Link Layer: A user can modify the MAC address of his WI-FI card to hide his identity.

Physical Layer: A physical layer attack, may be aimed at jamming transmission of a node or destroying a certain node's hardware

2.2.2.2. TYPES OF ATTACKS

There are several common methods used to attack ad-hoc networks. Some are similar to infrastructure networks while others are unique to ad-hoc networks. Note that these attack types include Physical Layer (jamming DoS), Link Layer (Spoofing), Network Layer (MIM), and Application Layer (Eavesdropping) [6].

a. Eavesdropping

Eavesdropping generally occurs when an unauthorized attacker listens to traffic on the network. In a wireless network, it is virtually impossible to prevent people from opening a radio receiver (hardware) and listening to what is being broadcast. [90] In addition, in an open network, a node could join and simply listen to traffic routed through that node. Eavesdropping can be used for listening to data traffic (what data one node is sending another) or to routing traffic (in an attempt to gain illegal access to the network). It is possible to prevent the attacker from gleaning useful information by employing encryption. Eavesdropping violates the confidentiality security measure listed in the threat model [6].

b. Man-in-the-middle Attack

Man-in-the-middle (MIM) attack involves an attacker forging his identity to two nodes that want to communicate. Each node thinks it is talking directly to the

other, while the MIM node reads, modifies or filters the data. MIM breaks confidentiality implicitly and authenticity and non-repudiation measures directly. The attacker could also use this position to affect integrity, availability and access measures. This type of attack can easily be carried out in an open network since any node can become a router. It could also be done at a lower level without joining the network.

c. Denial-of-Service Attack

Denial of service is an attack that simply prevents nodes from getting useful work done. This can occur at many different levels and can be passive or active. DOS attacks are theoretically unpreventable [90]. A simple example would be jamming the radio frequency used by the network, or nodes refusing to forward packets when asked to. Using secure routing protocols can mitigate the problems with the routing aspects of the attack, while hardware attacks can be reduced using spread spectrum [52] and similar technologies. DOS attacks break availability in the threat model.

d. Theft of resources

A node joins a network using a false identity and proceeds to consume resources that the node is not authorized to use. This would break the authenticity and access control and could implicitly break availability.

e. Corruption of data

A user corrupts data (in some subtle way) either at the hardware level using short bursts of false data or at the software level during routing. This is a subset of either DOS or MIM depending on the situation. This attack breaks integrity.

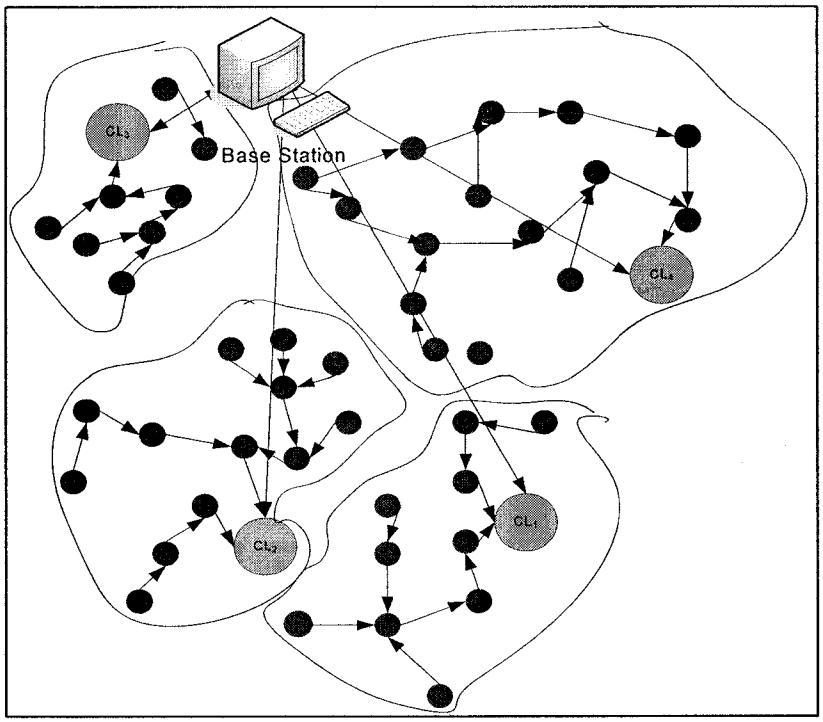

Fig. 3. A Sensor network with four clusters.

We direct the interested reader to a comprehensive study of different security attacks in networks to [37] and specifically for WAHNs to [51].

2.2.3. THREAT MODEL FOR SENSOR NETWORKS

Being wireless and working unattended mostly in a hostile environment renders sensor networks vulnerable to a myriad of attacks. Classical network attacks include eavesdropping, message interception, and message reproduction and jamming. In this thesis, we focus on attacks that aim to control the network permanently rather than causing a temporary disruption of the network functionality. We assume all bottom and second tier nodes to be subject to such attacks. Although we assume the base station to be unreachable to attackers, our solution should not depend heavily on that fact (i.e., we should avoid using the base station as a key distribution center (KDC) that shares a symmetric key with every individual node).

When nodes are captured, their memory can be read and erased or tampered with. Therefore, an adversary would know all the contents of a compromised node's memory. A widely accepted assumption [122], [14] is that an adversary will not launch an attack during the few minutes following the network initial deployment. Therefore, the network initialization will take place safely. Similar to LEAP [133], in DCK assumes that the adversary will not launch a coordinated attack i.e., if more than one sensor node were to be captured, a compromised node would not be aware of the location of other compromised ones unless they are its immediate neighbors. In order to handle collusion, any EBS-based collusion-resistant key assignment scheme like [46], might be used within DCK. Several dynamic keying schemes in cluster sensor network [122][14] assume that cluster leader nodes are less likely to be captured than regular sensor nodes. On contrary, DCK assumes other than base station every node is as likely to be captured as any other node. However, we assume at the same time that an adversary will not be able to distinguish between these a cluster leader and a sensor node either visually or through monitoring radio activity.

2.3. SECURE COMMUNICATIONS

Having a secure communication service is essential to most WAHN and WSN applications. In this section, we discuss different cryptographic solutions to provide secure communications in WAHNs and WSNs. We start with a classification of different solutions; we then discuss existing solutions in WHANs. Finally, we discuss existing solutions in WSNs.

2.3.1. CLASSIFICATION OF SECURITY SOLUTIONS

Security solutions for WAHNs and WSNs may be classified based on the communication model (e.g., Peer-to-Peer vs. Group-based), the implementation layer (i.e. physical, network, and application), and the technique used (cryptographic vs. non-cryptographic). Fig. **3** 4 shows the solution space. Non-cryptographic solutions may depend on techniques such as frequency hopping [58], while cryptographic solutions depend on data encryption. Physical layer solutions are hard to implement (e.g., frequency hopping). Network layer solutions may include secure routing, while

application layer solutions incorporate more sophisticated application logic (such as group membership, neighbors and surrounding environment awareness).

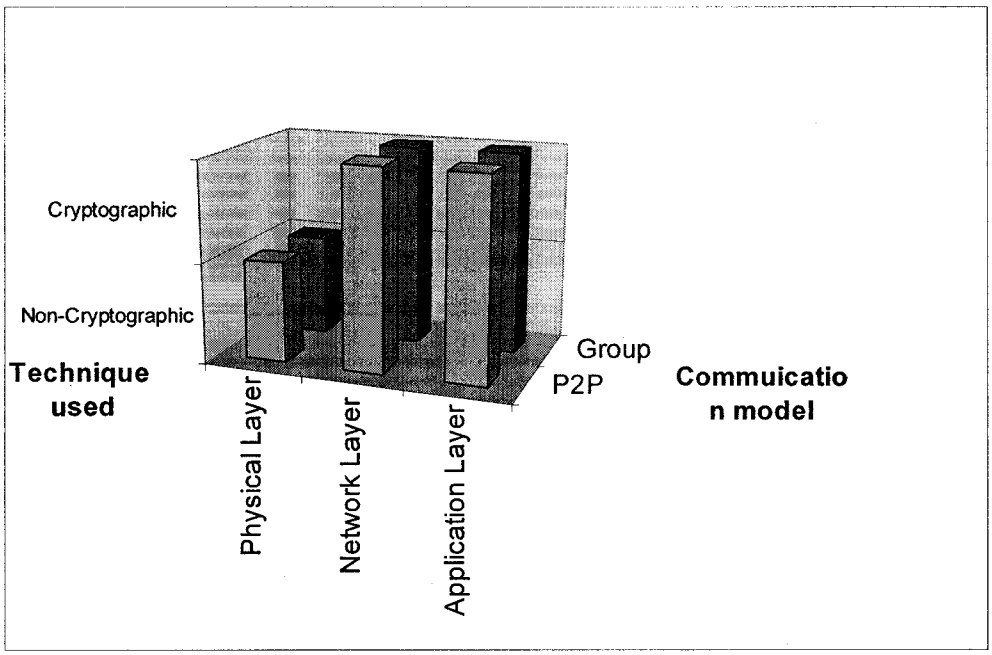

Fig. 4. Security solutions for WAHNs.

There are two major solution approaches to securing communications in WAHNs, a peer-to-peer approach and a group communications approach. In a peer-to-peer solution, nodes establish pair-wise connections using some shared key between each two communicating nodes. In the group communication approach, nodes participate in a securely communicating group whose members use a common group key to encrypt/decrypt group traffic.

Facilitating secure peer-to-peer communication on symmetric key cryptography in WAHNs has requirements similar to that of an on-line key distribution server (KDC) [134]. Thus, classical infrastructured networks schemes, such as such as Kerberos [61] and Otway-Rees [88], cannot fit in WAHNs. In order to overcome such problem, researchers have proposed several schemes such as [134] and [92].

Secure group communications requires the establishment of a communication group and distributing group key(s) to each group member. The simplest way is to have a *group key* used by the sender to encrypt data, and by the receivers to decrypt. Such a key is known as *Traffic Encryption Key (TEK)* [49]. The group key can be used as well in different security services such as authentication and maintenance of message integrity among users [113]. The major problem with group TEK is the group dynamics. However, in dynamic groups, membership changes cause the group key to be refreshed. Several group key management schemes have been used successfully in wired [61]networks such as Group Key Management Protocol (GKMP) [49] and several variations of Logical Key Hierarchy (LKH), [113] and [116]. The main problem with adapting such schemes in WAHNs is the lack of reliable multicast channel as well as the need for centralized group controller. In Section 6, several WAHN group key management protocols are discussed.

2.3.2. THE KEY MANAGEMENT PROCESS

One major issue with secure communications schemes in WAHNs is the use of several cryptographic keys for both traffic and control. Accordingly, managing these keys in a constrained environment, such as WAHN, is a major problem. In the next section, we discuss several components of key management and how they are performed in different secure communications schemes.

In general, rekeying can be considered to have a lifecycle (Fig. 5). The first step in such lifecycle is to establish the need for new or updated keys and determine which keys to be created or updated. Second, those keys need to be generated. The third step is to assign keys to users/parties/nodes, which are going to use them. The final step is to deliver the keys to these parties.

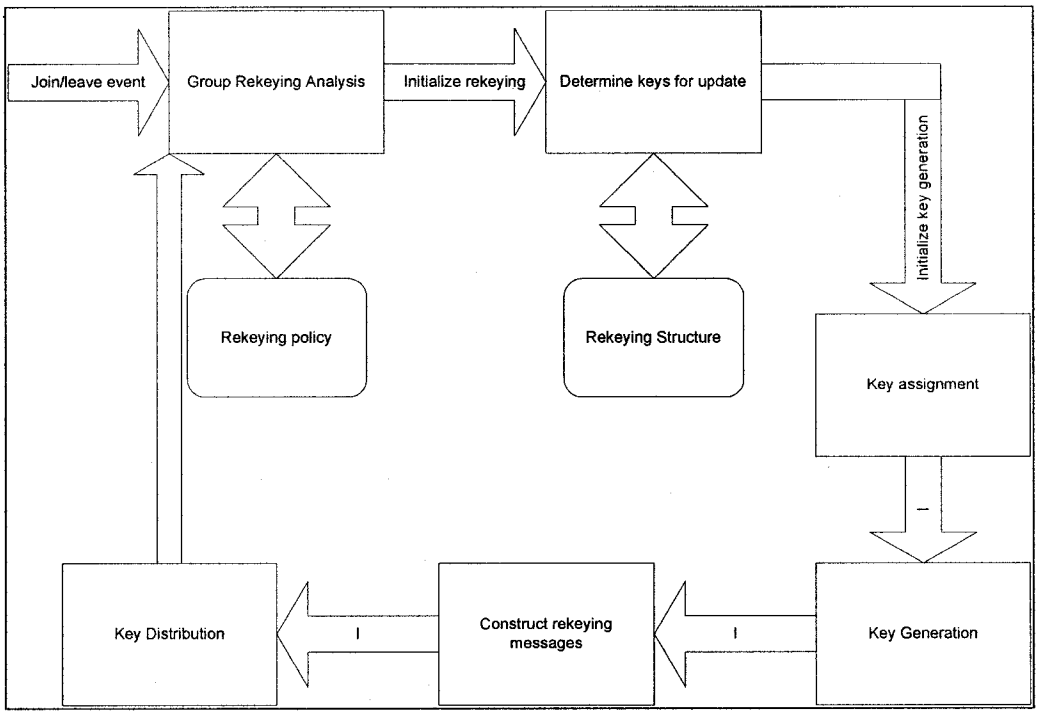

Fig. 5. Rekeying lifecycle in secure group communications.

In Fig. 6, we start with a system with secure communications established among all parties (e.g., WAHN nodes). A certain event (e.g., a new party joins or an existing party leaves) relevant to secure communication occurs. The system determines how to handle such an event by updating or generating a certain sub-set of the system keys (e.g., rekeying). The first step in handling such an event is key generation. New keys are generated to replace expired keys and then assigned to both rekeying messages and parties. The next step is to deliver these keys to appointed parties securely. Finally, by using the new keys, the system retains its original secure communication stage.

Although the term *key management* is used in the literature loosely as a synonym to rekeying (especially in the context of secure group communications and multicasting [116]), we use the term to describe all operations related to the maintenance of system

keys. In our abstract model, we define the following four key management processes in addition to the original rekeying.

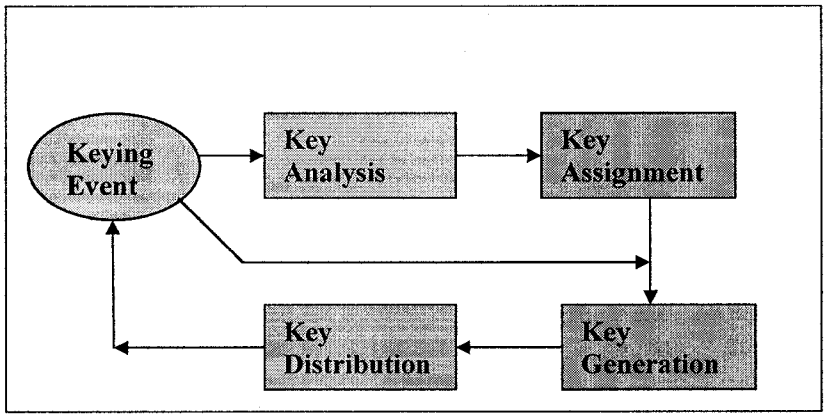

Fig. 6. The key management process.

Key analysis: keying requirements are analyzed to determine the required number of keys for the network as well as the number of keys needed by each node. Also, analysis may take place (using input from a detection system) to determine keys that needs update. Next, key assignment is performed. Actual keys are then generated (and encrypted) then distributed to their designated nodes.

Key assignment: refers to the mapping of keys to the different parties. Administrative key assignment is considered here since communication keys are simply assigned by agreement of parties wanting to establish a secure communication channel. Key assignment may be static (for example, each node is assigned the same set of keys throughout the network's lifetime) or dynamic depending on the key management solution employed.

Mapping decisions significantly impact the level of security offered by the key management scheme since a captured node may reveal all its keys to an attacker. If that node or a small number of nodes collectively possess all network administrative keys, capturing these nodes will jeopardize the security of the entire network. Therefore, when a node is captured, the fewer the number of keys known (or the more number of keys unknown) to that node, the greater is the reduction in security risk. However, since some

schemes depend on the existence of overlapping keys among nodes to establish communication among these nodes, decreasing the number of keys known to a node in these schemes may hamper network connectivity (see section 4 for security and performance analyses).

Random key assignment is the simplest and least expensive solution [32]. However, it limits control over the network security risk in case of multiple node captures. Deployment information (such as location or attack probability) have been used to reduce the attacker's probability to uncover more keys by capturing related nodes (for example. collocated nodes) [28][71].

Key generation: generation of administrative keys may take place once or multiple times over the lifespan of the network. The generation of communication keys is the responsibility of the communicating parties (i.e. sensor nodes, gateways, or base stations). In all cases, the key generating node(s) must be trusted by all key-receiving nodes. Keys might be as simple as bit string or as complex as a symmetric bi-variate polynomial [71].

In static key pre-distribution schemes, administrative keys are generated by a key server and loaded into nodes prior to deployment [16], [35]. In other schemes, new keys are generated regularly throughout the life time of the network [56], possibly with a different key generator at different times (as proposed in this article).

Key distribution (and re-distribution): refers to the delivery of keys to their designated nodes after they have been generated and assigned to the nodes. In case administrative keys are delivered to their destinations post network deployment, for example due to re-keying, other (previously distributed) administrative keys may be used to encrypt the new keys. The distribution of communication keys usually takes place after the network has been deployed. Communication keys are used for a short period of time and should be regularly updated (this may include analysis, assignment, generation and (re-)distribution).

2.3.3. KEY MANAGEMENT PROTOCOLS

Key management is an essential issue for secure communications in WAHNs. Key management protocols can be classified according to the communication model used

for secure group communications to be either pair-wise or group-based. In this sub-section, we discuss briefly both types of protocols. In Section 2.6, we discuss group key management protocols in greater detail.

2.3.3.1. PAIR-WISE KEY MANAGEMENT PROTOCOLS

In pair-wise node interactions, keys are redistributed to nodes during pre-deployment. Each pair of nodes share one secret key (either directly or indirectly) is used to establish an ephemeral session key whenever the two nodes decide to interact. As illustrated in Fig. 7, the two interacting nodes might seek help from other nodes to establish a secure path.

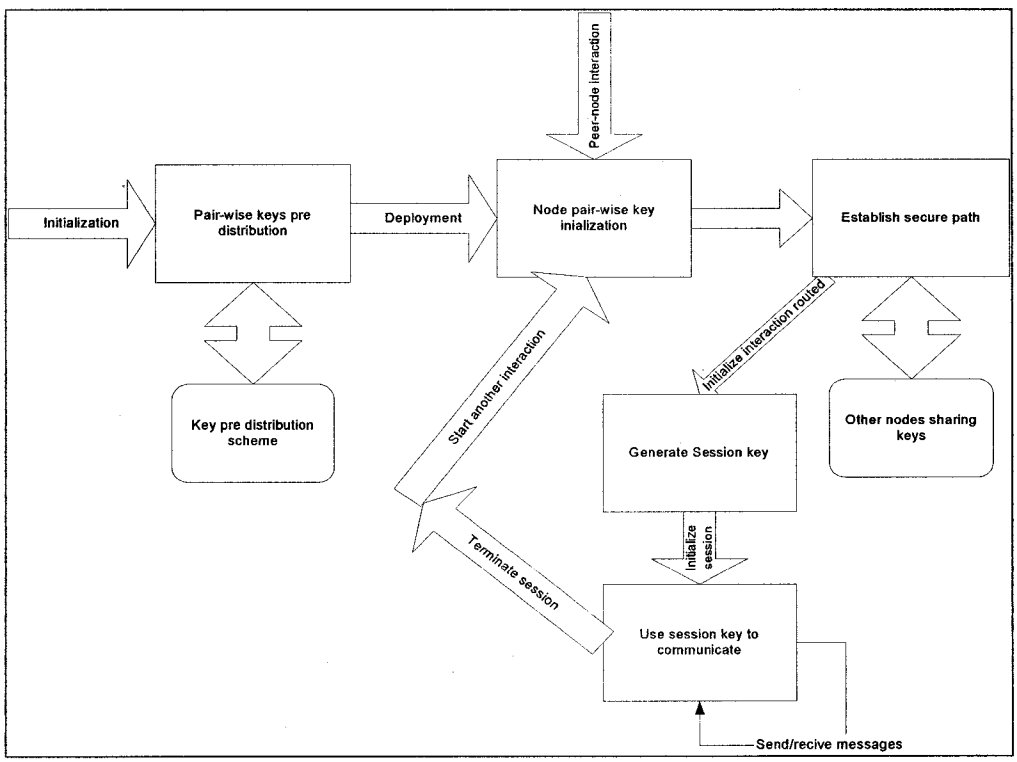

Fig. 7. Peer-to-peer communications using pair-wise keys.

For secure peer-to-peer communication between two mobile nodes in a wireless ad-hoc network it is necessary for the two nodes to share a secret key. This can be done using a public key infrastructure (PKI) [132]. Although there are several proposals for employing PKI in WAHNs, many ad-hoc networks cannot afford the deployment of such systems due to computational, communication, and storage constraints [11]. In a pair-wise key management system, shared keys are pre-distributed to nodes and are used to establish session keys using symmetric cryptography. In such schemes, key distributing and key assignment is performed in a pre-deployment fashion. The major advantage of pair-wise key management is full distribution with no need for a centralized Key Distribution Center (KDC) or online Certification Authority (CA). Examples of pair-wise key management protocols are Sensor Network Encryption Protocol SNEP [92] and the pair-wise key establishment protocol [132].

SNEP: Sensor Network Encryption Protocol (SNEP) [92] targets an environment of sensor nodes with few base-stations. However, the same protocol may be used for other types of WAHNs. SNEP focuses on how to use a shared secret key to exchange messages between two parties and provide confidentiality, data authentication, and integrity. Secret keys are assumed to be pre-distributed at a bootstrapping stage. In SNEP, each two communicating parties A and B are assumed to share a master secret key, X_{AB}. The two parties derive two independent pairs of keys from X_{AB} using a pseudo random function F. A pair of communication keys, K_{AB} *and* K_{BA}, is used for encrypting data back and forth. Another pair of MAC keys, K'_{AB} *and* K'_{BA}, is used for message hashing. Each party uses a local counter (e.g., C_A) to keep track of messages.

The pair-wise key establishment protocol: In [132], a probabilistic key sharing technique along with a threshold cryptography protocol to insure that each two parties exclusively share a certain key with an overwhelming probability. The protocols encounter three phases: *key pre-distribution phase, logical path establishment,* and *pair-wise key establishments.*

The *key pre-distribution* phase includes both key distribution and key assignment. In the *key pre-distribution*, the off-line key server loads each node u with m distinct keys from

the key pool P of l keys $\{k1; k2; : : : ; kl\}$ prior to the formation of the ad-hoc network. As a result, each key in the key pool has a probability of m/l to be chosen by each node. This procedure assigns to each node a set of m keys (out of l) whose IDs can be deterministically obtained using the node ID.

The *path establishment* procedure is executed when a node wants to securely exchange messages with other nodes in the network even if they do not directly share a key. Two nodes can establish a logical path through other nodes with which they share other keys. A node u can independently compute Iv, the set of key ids corresponding to a node v's key set. Therefore, without proactively exchanging the set of its key ids with others, a node knowing the ids of its l neighbors can determine which two neighbors share which keys. Two arbitrary nodes u and v, node u can use a proxy node x as follows:

$U \rightarrow x : \{M\}_{kux}$

$x \rightarrow u : \{M\}_{kxv}$

$u \rightarrow v : \{M\}_{kxv}$

2.3.3.2. CLASSIFICATION OF GROUP KEY MANAGEMENT PROTOCOLS

Key management plays an important role in enforcing access control on the group key (and consequently on the group communications). It supports the establishment and maintenance of key relationships between valid parties according to a security policy being enforced on the group [74]. It encompasses techniques and procedures that can carry out member identification and authentication, access control, and generation, distribution and installation of key material. According to [98], key management schemes can be broadly classified into three major categories.

- **Centralized group key management protocols**

  A single entity, e.g. KDC, is employed for controlling the whole group; hence a group key management protocol seeks to minimize storage requirements, computational power on both client and server sides, and bandwidth utilization.

- **Decentralized key management protocols.**

  The management of a large group is divided among subgroup managers, trying to minimize the problem of concentrating the work in a single place.

- **Distributed key management protocols**

  There is no explicit KDC, and the members themselves do the key management. All members can perform access control. The generation can be either contributory, meaning that all members contribute some information to generate the group key, or done by one of the members.

Key management protocols may be also classified based on the structures used for rekeying. **Simple (or linear) rekeying** structures involve the use of a single key per user, and distributing the TEK using such key. Example of such systems is GKMP [49]. Graph-based systems uses a key graph, Iolus [78], or key tree Logical Key Hierarchy (LKH) [116] and [113]. Other systems use a combinatorial structure such as Exclusion Basis System (EBS) [30]. In Section 2.6, we discuss group key management protocols in more detail.

2.4. GROUP KEY MANAGEMENT PROTOCOLS

Recently a number of key management schemes have been developed for sensor networks. Examples include [25][122][14][121][17]. Key management schemes for sensor networks may be broadly classified into static and dynamic keying based on whether the administrative keys (those used to establish communication keys) are updated or not after the initial network deployment and set up. A well known static keying scheme is due to *Eschenauer* and *Gligor* [35]. In this scheme, each sensor node is assigned k keys out of a large pool P of keys in the pre-deployment phase. Neighboring node may establish a secure link only if they share al least one key, which is provided with a certain probability based on the selection of k and P. A major advantage of this scheme is the exclusion of the base station in key management. However, successive node captures enable the attacker to reveal network keys and use them to attack other nodes. An enhancement to such scheme was proposed in [17] in which two nodes can establish a link only if they share q keys. Liu and Ning [70] provided further enhancement by using t-degree bi-variate key polynomial. Since an attacker needs to capture at least $t+1$ nodes to obtain any *t-degree* polynomial, this solution was shown to significantly enhance network resilience to node capture.

On the other hand, dynamic keying schemes change all keys revealed to an attacker upon node capture. The major advantage of dynamic keying is enhanced network survivability since any captured keys are replaced in a timely manner.

Also, when adding new nodes, unlike static keying, the probability of network capture does not necessarily increase. Jolly et al. [56] have proposed a key management scheme that is based on the Identity Based Symmetric Keying. Although their approach requires very few keys to be stored at each node, the re-keying procedure is inefficient due to the large number of messages exchanged for key renewals. In addition, they require a centralized key server to play a major role in key management. Another group of dynamic keying schemes are based upon Exclusion Basis Systems (EBS) - a combinatorial formulation of group key management problem. In EBS-based schemes, each node is assigned k keys out of pool of size $k+m$ keys. Once one or more nodes are captured (or suspected to be captured), rekeying takes place by generating replacement keys, encrypting them with all the m keys unknown to the captured nodes and distributing them to the other nodes. However, since the value of m is selected to be relatively small, to make rekeying feasible in terms of number of messages, a small number of nodes may collude and collectively reveal all the network keys. EBS-based schemes with collusion-resistance have been proposed recently in [14] and [121]. While more energy efficient than *Jolly et al's* scheme, both schemes still considerably rely on a centralized key server to perform rekeying.

In this section, we discuss factors that affect the selection of a specific group key management solution as well as different existing types of key management schemes with some examples.

2.4.1. FACTORS AFFECTING GROUP KEY MANAGEMENT SOLUTIONS

Group key management protocols attracted many researchers through the last decade (e.g. [115], [24], and [30]). Most of existing protocols were proposed with the Internet in mind as a deployment environment. Researchers focused on performance issues such as storage and communication cost, based on the assumption of having IP-Multicast. The situation is different in WAHNs. Wireless communications is much less

reliable, multicast might not be supported, and group size may grow up to several tens of thousands or even millions. Adopting one of these protocols or even developing a new protocol suitable for WAHNs should take several factors into consideration. Some factors are related to the group issues, others may be related to network characteristics, application environment, or performance constraints. In the following sub-sections, we discuss some of these factors.

2.4.1.1. GROUP-RELATED FACTORS

The major factor concerning a group is the group size. In some applications, such as M-Commerce or rescue missions, group size is kept within tens of nodes. In other applications, such as data aggregation in smart sensor networks, group size may reach several ten or hundreds of thousands of nodes [111]. A group key management protocol should be scalable enough to support such large group sizes while keeping the communication and storage overhead as low as possible.

Group dynamics is another major concern. According to [49] group Traffic Encryption Key (TEK) (and other administrative keys), where to changed every time a node joins or leaves the group. The rate of nodes joining and leaving the group significantly affects the performance of any group key management protocol. To reduce such overhead, researchers have proposed batch rekeying rather than individual rekeying [65]. However, batch rekeying may cause the system to be vulnerable for short periods of times in which certain nodes have left the group while rekeying did not take place yet.

Group structure and the capability of sub-grouping is another important issue. In groups with large number of nodes, having a flat group with rekeying taking place by distributing new keys to thousand of nodes through unreliable communications, is very undesirable. It is an important advantage for a key management protocol to have the ability of sub-grouping to avoid such a huge overhead. However, classical sub-grouping protocols with multiple encryptions, such as Iolus [78] or Dual Encryption Protocol [24] introduce unacceptable overhead. There might be a need for new protocols that allows sub-grouping without the cost of multiple encryptions. Combinatorial-based protocols are promising in that area [30].

2.4.1.2. NETWORK-RELATED FACTORS

The WAHN environment is quite different from conventional wired networks. Those differences significantly affect the applicability of key management protocols of conventional networks to WAHNs. WAHNs have less reliable communications, less support for multicast, different configuration and networks structure, as well as the factor of mobility.

Most existing key management protocols were developed to target the Internet with IP-Multicast. In WAHNs, the network environment is quite different. Multicasting protocols in the Internet generally focus on optimizing either the distance (or cost) from the sender to each receiver individually, or the total cost of the multicast tree. Multicasting in ad-hoc networks is more challenging because of the need to optimize the use of several resources simultaneously [48]. Among those hurdles are the limited battery power at each node, the lack of a centralized status, and the varying link quality between mobile nodes that act as routers [48]. Accordingly, in some situations, key management and group communications in general may depended on a network level unicast service and implement the multicast at the application layer. Examples include AMRoute protocol [117] and PAST-DM [44]. Some recent protocols, such as EKDMS, do not assume a network-level multicast support and instead use Application Level Multicast (ALM) as an alternate.

WAHNs may have either flat or clustered structure. In WAHN applications with flat structure, nodes are autonomous and enjoy similar capabilities [15]. Nodes may be geographically collocated and any two arbitrary nodes may communicate via the same multi-hop channel. In some other applications with clustered or hybrid network structure, some nodes (super-nodes) may have special communications or storage capabilities. An example of such nodes is the base stations in sensor networks [110]. Nodes may be grouped in clusters (either geographically or logically) with super-nodes performing inter-cluster communications. Other applications may also integrate other non-WAHN networks (such as cellular networks) with a different communication model, see CAMA [9].

Mobility is a major difference between WAHNs (and specifically MANETs) and other networks. From a group communications point of view, intense mobility of WAHN

nodes may cause frequent disconnections and/or leaves. This may cause nodes to miss rekeying messages and/or data streams. Solutions to this problem may include a threshold-based scheme for rekeying messages such that a node needs to collect a threshold number, t, out of the total number of messages, n, in order to reconstruct all the rekeying material. Examples include the use of Forward Error Correction (FEC) [128], simple t-out-of N simple *exclusive OR* [69], and *Verifiable Secret Sharing* (VSS) [42].

2.4.2. SECURITY AND PERFORMANCE REQUIREMENTS

Different WAHN applications have different security and performance requirements. Characterizing such requirements is essential for selecting or developing a key management protocol for WAHNs. Security requirements include the providing forward and/or backward secrecy; the need for authentication, the level of trust among nodes, the possibility of collusion, and the maximum vulnerability period the system can go through [98]. Performance constraints include storage requirements for keys, communications efficiency, as well as scalability [65].

Rekeying is the primary solution to providing both backward and forward secrecy. However, some applications do not mandate backward secrecy, such as time varying data sensing where data is offered as public after a short time. In such case, a key management protocol should take advantage of releasing such a constraint by reducing the rekeying overhead. Some systems sacrifice forward secrecy for performance tradeoff such as GKMP [49].

The existence of an authentication service is a non-essential requirement that may not be needed in certain applications (e.g., open networks). In such a case, group management protocol should accommodate more dynamics by waiving such a requirement. Nodes in WAHN are generally autonomous. In open WAHNs applications, any node can join the group, especially in the absence of an authentication service. In such a case, the trust level among nodes should be minimal and extra verification techniques (e.g., Verifiable Secret Sharing) should be imposed. In other applications, the authentication service sets a high level of trust among nodes so that no verification overhead should be incurred.

Collusion may be defined as the case in which several nodes can come together and collectively reveal all administrative keys and breaks both forward and backward secrecy.

The possibility of such an event depends on the ease of communications among nodes as well as the amount of administrative information acquired by each node. Some decentralized systems are subject to collusion such as VERSA [112]. Although combinatorial key management protocols use a fewer number of keys to manage a large number of users, they are subject to collusion [30]. Some distributed combinatorial protocols use collusion-resistant key assignment.

Due to the scarce resources in WAHNs, efficient storage and communications is essential to key management protocols. In wired networks, researchers focused on optimizing the number of rekeying messages as well as the number of keys stored. However, with the assumption of IP-Multicast support, communication cost was quantified in terms of number of multicast messages. In WAHNs, multicast may be either expensive or even not supported at all. In such case, communication cost should be optimized further taking in consideration both the ability of limited broadcast as well as unicast-based rekeying,. Such solutions should also maintain a lower storage cost per node to achieve scalability in large group sized WAHNs applications. Combinatorial key management protocols [30] were shown to require number of keys comparatively smaller than tree-based solution for the same number of nodes. Communication cost includes number of control messages (unicast and multicast) exchanged upon join or leave as the group continues. In batch rekeying, the communication cost an be measured in number of messages exchanged upon rekeying, taking in consideration the rekeying rate ore threshold [118].

2.5. GROUP KEY MANAGEMENT PROTOCOLS IN WSNS

In this section, we briefly introduce the network and attack models, as well as some static and dynamic key management schemes that we consider for comparison.

2.5.1. MODEL ASSUMPTIONS

We consider a typical sensor network comprised of a large number of resource-limited sensor nodes deployed in a field. A resource-rich base station is deployed alongside the sensor nodes for communication with the outside world [1]. Each sensor node stores keying material that is injected into nodes either through pre-distribution or during a post-deployment bootstrapping phase, (which is assumed to be secure). Secure

communications among network nodes is maintained using encryption keys generated using this keying material.

Being wireless and working unattended mostly in a hostile environment renders sensor networks vulnerable to a myriad of attacks. Classical network attacks include eavesdropping, message interception, and message reproduction and jamming. In this thesis, we focus on attacks that aim to control the network permanently rather than causing a temporary disruption of the network functionality. When nodes are captured, their memory can be read and erased or tampered with. Therefore, an adversary would know all the contents of a compromised node's memory, including keying material. A widely accepted assumption is that an adversary will not launch an attack during the few minutes following the network initial deployment. Therefore, the network initialization will take place safely. Node capture attacks can be either simple or coordinated. In a simple attack, if more than one sensor node were to be captured, a compromised node would not be aware of the location of other compromised ones unless they are its immediate neighbors. On the other hand, in a coordinated attack, an attacker can foster collusion among nodes that are not co-located, and thus the network topology is not a factor in the success or failure of the attack. Most dynamic key management schemes assume the simple attack models, while most static key management schemes assume the coordinated attack model. In our proposed model, we assume a coordinated attack with an upper limit on the rate at which the attacker can capture network nodes.

2.5.2. STATIC KEYING SCHEMES

Recall that static keying schemes assume administrative keys once pre-deployed in the nodes, will not be changed. The basis of most existing key management schemes have originated in the key pre-distribution scheme proposed by Eschenauer and Gligore in [35]. This scheme selects k keys for each node out of a large pool P. A major advantage of such scheme is incurring no post-deployment communication overhead on sensor nodes (for administrative keys). However, successive node capture enables the attacker to reveal keys stored in the nodes and use them to attack other nodes. The authors show that on average half of the keys are used to secure links between nodes, and thus, successive node capturing hampers the network survivability. A major advantage of

this scheme is the exclusion of the base station in key management. An enhancement of this scheme has been proposed in [8] in which two nodes can establish a link only if they share q keys.

Liu and Ning [70] provided further enhancements by using *t-degree* bi-variate key polynomials. Instead of selecting k keys for each node as in the basic probabilistic scheme, each node carries k bi-variate polynomials. Each is stored as $t+1$ shares. Two nodes can directly communicate only if they have at least one shared polynomial. Since an attacker needs to capture at least $t+1$ nodes to obtain any *t-degree* polynomial, this solution was shown to significantly enhance network resilience to node capture as long as the number of captured nodes is below a certain threshold (around 3% as shown in [56]). However, if the number of captured nodes exceeds this threshold, the number of keys revealed to the attackers jumps sharply to reach almost 100%.

Another issue with the abovementioned static schemes is the reduction in network connectivity resulting from the use of a large key pool, P. As P increases for the same k, network connectivity decreases since only nodes that overlap in one or more keys (or key polynomials) can directly interact.

2.5.3. DYNAMIC KEYING SCHEMES

Dynamic keying schemes change administrative keys revealed to an attacker upon detection of node capture. Administrative keys can also be changed periodically or on demand to increase network resilience to capture. The major advantage of dynamic keying is enhanced network survivability since any captured keys are replaced in a timely manner. Also, upon adding new nodes, unlike static keying, the probability of network capture does not necessarily increase.

An example of dynamic keying schemes is due to Jolly et al. [56] who proposed a key management scheme that is based on the Identity Based Symmetric Keying. Although their approach requires very few keys to be stored at each node, the re-keying procedure is inefficient due to the large number of messages exchanged for key renewals. In addition, they require a centralized key server to play a major role in key management.

Another group of dynamic keying schemes are based upon Exclusion Basis Systems (EBS) - a combinatorial formulation of group key management problem developed by

Eltoweissy et al in [32]. In EBS-based schemes, each node is assigned k keys out of pool of size $k+m$ keys. Once one or more nodes are captured (or suspected to be captured), rekeying takes place by generating replacement keys, encrypting them with all the m keys unknown to the captured nodes and then distributing them to the other nodes. However, since the value of m is selected to be relatively small, to make rekeying feasible in terms of number of messages, a small number of nodes may collude and collectively reveal all the network keys. A brief description of the EBS methodology is included in Section 3 for clarity. For details on EBS, please refer to [30].

Eltoweissy et al also proposed the use of EBS key management in a sensor network with a virtual infrastructure [32]. In their scheme, the sensor network efficiently establishes a coordinate system around the base station. All nodes located in the same coordinate cell are assigned the same EBS key combination and are considered collectively as an EBS group member. Rekeying takes place on the cell level by sending m messages of k keys each to evict nodes in a specific cell. Although very efficient, this scheme does not address collusion and assume coarse keying granularity due to the assumption of ID-less nodes..

EBS-based schemes with collusion-resistance have been proposed recently by Younis et al in [121] and Chorzempa et al in [14]. While more energy efficient than Jolly et al's scheme, both schemes still considerably rely on a centralized key server to perform rekeying.

As stated earlier, in dynamic keying, mitigating key compromises (by capturing nodes) is handled by rekeying. In EBS-based schemes, all k keys known by the captured node are replaced with updated k keys, encrypted with m keys unknown to the attacked node and broadcasted to the network. If the attacker can capture and foster the collusion of enough nodes that collectively know the entire set of $k+m$ keys (we call the set of captured nodes in this case the collusion chain), the network is considered compromised. In [124], the authors proposed a location-based key assignment to increase the length of the collusion chain compared to random key assignment.

In our study, we consider the Liu and Ning scheme [70] as an example of static schemes and Younis et al. scheme [123] as an example of dynamic keying.

2.6. COMBINATORIAL GROUP KEY MANAGEMENT PROTOCOLS

Combinatorial group key management protocols depend on a different key structure, a combinatorial matrix. A set of (k+m) administrative keys is used to support a set of N multicast users. The administrative keys are distributed among users such that each user knows a different set of keys. Exclusion Basis System (EBS) [30] is an example of a centralized combinatorial protocol. Efficient Decentralized Key Management Scheme is still needed.

2.6.1. EXCLUSION BASIS SYSTEM (EBS)

Eltoweissy *et al* introduced an Exclusion-Basis System (EBS) framework for the efficient management of keys in secure group communications [30]. The EBS framework uses a combinatorial formulation to maintain administrative keys (keys used to securely distribute session encryption keys). An EBS is defined as a collection Γ of subsets of the set of members. Each subset corresponds to a key and the elements of a subset $A \in \Gamma$ are the nodes that have that key. An EBS Γ of dimension (n, k, m) represents a situation in a

TABLE 1.

The canonical matrix A for EBS(10, 3, 2)

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10
K1	1	1	1	1	1	1	0	0	0	0
K2	1	1	1	0	0	0	1	1	1	0
K3	1	0	0	1	1	0	1	1	0	1
K4	0	1	0	1	0	1	1	0	1	1
K5	0	0	1	0	1	1	0	1	1	1

secure group where there are n members numbered 1 through n, and where a key server holds a distinct key for each subset in Γ. In this section, we will use the terms "key" and "subset" interchangeably. If the subset A_i is in Γ, then the key A_i is known by each of the members whose number appears in the subset A_i. Furthermore, for each $t \in [1,n]$ there are m elements in Γ whose union is $[1,n]-\{t\}$. From this it follows that the key server can evict any member t, re-key, and l*et al*l remaining members know the replacement keys for the k keys they are entitled to know, by multicasting m messages encrypted by the keys corresponding to the m elements in Γ whose union is $[1,n]-\{t\}$. Each new key is encrypted by its predecessor to limit decipherability only to the appropriate members.

To construct $EBS(n,k,m)$ for feasible n,k and m, a canonical enumeration of all possible ways of forming subsets of k objects from a set of $k+m$ objects may be employed. As in [30], we choose an enumeration where each element of the sequence is a bit string of length $k+m$, where a 1 in the i^{th} position of a string means that object i is included in that subset, for all i $(1 \le i \le k+m)$. Note that, every bit string in this enumeration will have exactly k ones. We use a canonical type of enumeration for the binomial coefficient $\binom{k+m}{k}$ subsets using induction on $k+m$. For any k and m, let $Canonical(k,m)$ be the canonical enumeration of all $\binom{k+m}{k}$ ways to form a subset of k elements from a set of $k+m$ objects. For the sequence of bit strings in $Canonical(k,m)$, a matrix A is formed, where k and m are understood, and whose $\binom{k+m}{k}$ columns are the successive bit strings of $k+m$ length, each with k ones. A is called the canonical matrix for $EBS(n,k,m)$. For example, the canonical matrix A for *EBS(10, 3, 2)* contains the enumeration of all $C(5,3)$ ways to form a subset of 3 keys from 5 keys, A is shown in Table 1.

In [30], Eltoweissy *et al* note that all systems for key management are examples of an EBS. That is, given a logical system for key management, one can describe each key as a

subset of users, i.e. those users that possess that key. Eltoweissy *et al* show how a binary key management tree is an EBS. The overhead of an optimum compact EBS (one where $k+m$ is equal to the number of keys in the system) is half that of a binary key tree [30]. In a collusion-free environment, using EBS for key management guarantees forward and backward secrecy.

A compact EBS may suffer from collusion attacks that involve a small number of evicted users. In [30], some solutions to collusion resistance in EBS are proposed. The selection of certain values of the parameters k and m might increase the overlap between keys known by different users. We plan to propose a better solution that depends on appropriately selecting k and m.

2.7. SUMMARY

EBS has been used as a component for many key management solutions in both WAHNs and WSNs. These solutions have several problems.

In Ad-hoc networks, the problems include:

Duplication of group management functions [73]. In current architectures, the multicast session and group membership is maintained at the network layer (for all group-based applications). However, the group security and key management are performed at the application layer. Clearly, in this approach, there is significant computational overhead and a need for consistency (between application and network layers) management. In addition, network layers in WAHNs cannot implement the multicast functionality as efficiently as the Internet. For these two reasons, we propose to combine the multicast group functionality and the key management functionality at the application layer itself.

Overhead of distributing control traffic. Current key management protocols use the network level multicast channel to distribute both data and rekeying/control traffic [12], [116],[118]. Accordingly, all group members unnecessarily receive all rekeying messages. In the proposed architecture, we avoid this overhead by ensuring that the keys are distributed only to the required nodes. This is possible due to the integration of multicast and key management functions in the application layer.

Scalability. IP-multicast has a scalability problem with respect to the number of applications or groups that can be supported [27]. The scalability issue is the main motivation behind shifting to ALM. In our architecture, we assume no network-level multicast support in WAHNs and use application-level multicast. ALM provides more scalability with less overhead for larger number of applications with limited group membership.

Distribution of key management functions. In most existing key management protocols, a centralized group controller maintains all the key management functionality [12], [116], [118]. In a limited resource environment, such as WAHNs, this constitutes an unacceptable overhead and drain of resources for a single node. Moreover, WAHN nodes tend to be unreliable and are prone to failure. The novel architecture handles this problem by having separate modules for key generation, assignment and distribution, each performed in a fully-distributed and fault-tolerant fashion.

Handling miss-behaving and untrusted nodes. WAHN nodes are autonomous and each node may choose whether or not to participate in group management [73]. In addition, nodes may falsify information or collude by exchanging rekeying material [30]. In the novel architecture, the key management modules take these threats into consideration by applying techniques such as verifiable secret sharing, threshold-bases key generation, and collusion-resistant key assignment.

In Wireless Sensor networks, the list further includes: A large number of keys need to be managed in order to encrypt and authenticate all sensitive data exchanged. However, due to the characteristics of sensor networks, including lack of physical protection and the resource constrained nature of sensors and sensor networks, most existing key management solutions developed for other networks may not be feasible for sensor networks (for example. PKI-based solutions). The tradeoff between managing acceptable levels of security and conserving network energy for sensor network operation is a challenging task.

In this thesis, we propose an efficient dynamic key management scheme using key polynomials that:

(1) Achieves and maintain high network connectivity required to establish efficient communication paths;

(2) Performs efficient rekeying as needed to enhance network resilience to attacks. We compare our scheme to other leading static and dynamic schemes with respect to collusion resistance, overhead, and connectivity.

The objective of this thesis is to propose an efficient key management solution for both WAHNs and WSNs that overcomes these problems. In the next chapter we describe our proposed solution for WAHNs.

CHAPTER III

KEY MANAGEMENT IN WIRELESS AD-HOC NETWORKS

Secure communication is essential for the acceptance and broader adoption of Wireless Ad-hoc Networks (WAHNs) in information-sensitive applications such as mobile commerce, emergency medical assistance, ad-hoc conferences, and tactical defense operations [15]. Typically, WAHNs comprise of autonomous end-nodes that must collaborate for end-to-end connectivity. Therefore, communication in WAHNs is inherently collaborative requiring group communication. In addition, secure group communication is also needed to support WAHN applications such as collaborative team investigations, mobile auctioning, and first-responder support [54].

WAHNs exhibit distinguishing characteristics that limit the applicability of contemporary architectural solutions designed for infrastructure networks. Here, nodes work together to setup an ad-hoc network that does not rely on a physical networking infrastructure. Communications are performed via multi-hop wireless broadcast transmission among nodes with limited transmission range. The lack of physical protection of communications together with the constrained capabilities of small wireless devices increases the likelihood of failure [31]. Node autonomy in open WAHNs further complicates the problem; un-trusted nodes may falsify information; selfish nodes may refuse to cooperate. Moreover, a powerful mobile external adversary may compromise one or more nodes and use them to launch different attacks that might hamper the network functionality.

To accommodate these special needs of WAHN environments, we propose a multilayered architecture for secure group communication. Our architecture does not assume the availability of any network-level support for multicast. Instead, it integrates secure group management and multicast functions, and places them in the application layer.

Among the several modules of the proposed architecture, group management is especially sensitive to the characteristics of the underlying infrastructure. While several key management protocols such as Group Key Management Protocol (GKMP) [49]), Logical Key Hierarchy (LKH) [113], and One-way Function Chain Tree (OFCT) [12] have been

proposed in literature, we find that they are not directly transferable to WAHN environments. Here, we propose and integrate schemes for different group management components (e.g., key generation, key assignment, and key distribution). We also show how each component supports security, fault–tolerance, and efficiency.

The chapter is organized as follows. Section 1 presents our architecture. Sections 2 to 4 describe the design issues of different components to implement the proposed architecture. Section 2 discusses the key distribution. Section 3 describes the proposed group generation. Section 4 introduces our proposed key assignment scheme that minimizes possible collusion among nodes.

3.1. OVERVIEW OF THE PROPOSED ARCHITECTURE

In this section we discuss the need for a new architecture, our novel architecture, and its feasibility and practicality.

3.1.1. NEED FOR A NEW ARCHITECTURE FOR WAHNS

As explained in the introduction, WAHNs have some special characteristics that demand new architectural paradigms. Here, we briefly summarize some of the key issues and discuss how the novel architecture handles it. We here show how to address the problems identified in section 2.7.

Avoiding duplication of group management functions [69]: In current architectures, the multicast session and group membership is maintained at the network layer (for all group-based applications). However, the group security and key management are performed at the application layer. Clearly, in this approach, there is significant computational overhead and a need for consistency (between application and network layers) management. In addition, network layers in WAHNs cannot implement the multicast functionality as efficiently as the Internet. For these two reasons, we propose to combine the multicast group functionality and the key management functionality at the application layer itself.

Avoiding overhead of distributing control traffic: Current key management protocols use the network level multicast channel to distribute both data and

rekeying/control traffic [12],[116],[118]. Accordingly, all group members unnecessarily receive all rekeying messages. In the proposed architecture, we avoid this overhead by ensuring that the keys are distributed only to the required nodes. This is possible due to the integration of multicast and key management functions in the application layer.

Scalability, flexibility, and application semantics: As described in Section 2, the major attractions to shift the multicast functionality to the application layer are (a) Scalability with respect to group size, number of groups as well as frequency of membership change [27]. (b) Flexibility of the overlay structure to adapt dynamically to membership and topology change [66] (e.g. mobility). (c) The ability to integrate other functionalities with the multicast (e.g. key and group management in our case) [66].

Distribution of group and key management functions: In most existing key management protocols, a centralized group controller maintains all the key management functionality [12] [116] [118]. In a limited resource environment, such as WAHNs, this constitutes an unacceptable overhead and drain of resources for a single node. Moreover, WAHN nodes tend to be unreliable and are prone to failure. The novel architecture handles this problem by having separate modules for key generation, assignment and distribution, each performed in a fully-distributed and fault-tolerant fashion.

Security and trust: WAHN nodes are autonomous and each node may choose whether or not to participate in group management [69]. In addition, nodes may falsify information or collude by exchanging rekeying material [30]. Moreover, an attacker might hijack one or more nodes and use them to reveal all the network keys. In the novel architecture, the key management modules take these threats into consideration by applying techniques such as verifiable secret sharing, threshold-based key generation, and collusion-resistant key assignment.

3.1.2. THE NOVEL ARCHITECTURE

The proposed novel architecture is shown in Fig. 8. It consists of four layers (similar to the ISO OSI reference model). The top layer is the application layer where group applications (e.g., secure conferencing, secure mobile auctions, etc.) reside and execute. Below this layer is the sub-application layer that supports group applications with secure group communication primitives as well as flexibility and security mechanisms to conform to both trust among network nodes as well as application semantics. Below this layer is the network layer which is assumed to offer only unicast packet delivery (due to scalability considerations in WAHNs). The bottom most layer is the physical layer which is a standard wireless MAC layer (e.g., 802.11).

Except for the sub-application layer, which is introduced in the novel architecture, the remainder of the layers correspond to the similar ones in standard wireless ISO OSI stack. In the rest of the chapter, we focus on the newly introduced sub-application layer.

The sub-application layer integrates multicast communication with secure group management. One or more group applications run at the application layer utilizing the secure group communication services provided by the underlying layer. The sub-application layer consists of three key modules: authenticator, group manager, and Application-Level Multicast (ALM) module. The interaction among these modules is well explained through the following scenario.

Upon receiving a join request from a user node, the authenticator evaluates the credentials of the node and decides whether or not to honor the request. After authenticating the source of the request, the authenticator forwards the request to the group manager, which adds the new node to the multicast group and performs rekeying to guarantee backward secrecy [98]. Rekeying performed both on demand as well as on regular basis to guarantee backward secrecy [118]. The ALM module provides a multicast primitive at the application layer to deliver messages through underlying network layer (via unicast) to group members. The group manager utilizes this service to deliver both data and rekeying material to relevant users.

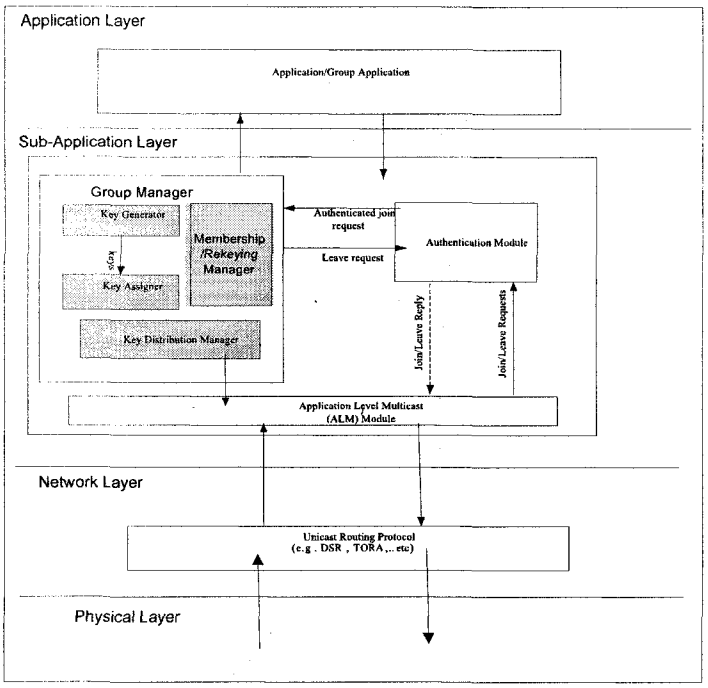

Fig. 8. A Novel architecture for secure group communications in WAHNs.

Among the three modules, the group manager architecture is unique and is designed to work in the WAHN environments without network level multicast support. It consists of the following four components, which perform their functions in a distributed, fault-tolerant, and efficient fashion. Each WAHN node may run these modules.

Membership/Rekeying Manager is responsible for reporting leaves and joins by current members and initializes rekeying according to these membership changes. It is also responsible for reporting such membership changes to the ALM module in order to modify the multicast overlay structure.

Key Generator is responsible for generating new keys for joining members as well as modified versions of current keys for rekeying to preserve forward secrecy.

Key Assigner is responsible for assigning current and newly generated keys to group members in order to maintain current secrecy and minimize the possibility of collusion.

Key Distribution Manager is responsible for distributing new keys at each rekeying time to appropriate members either through individual unicast or ALM messages.

The overall structure of the group manager is shown in Fig. 8. In the following, we overview each of the components in the structure and show how they support the overall design objectives of the novel architecture.

3.1.3. IMPLEMENTATION FEASIBILITY

We now look at the implementability issues of the novel architecture. As mentioned above, the physical layer, the network layer, and the application layer are standard layers and do not need further explanation. So we concentrate on the implement ability of the sub-application layer and its three components.

Since the authenticator is responsible for evaluating the credentials presented by joining members, existing authentication protocols based on PKI should be adequate. However, contemporary PKI protocols designed specifically for WAHNs (e.g., MOCA [126]),

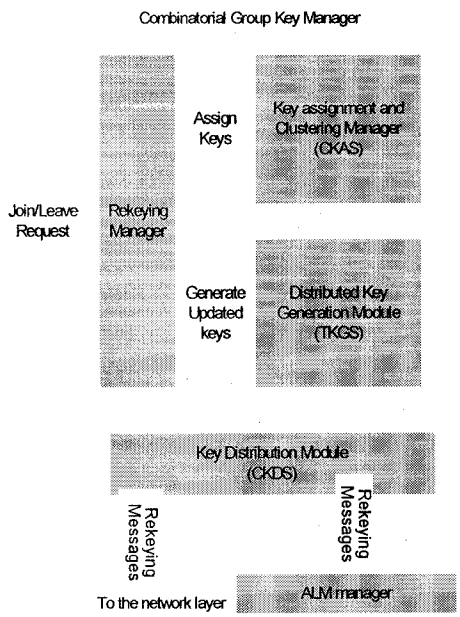

Fig. 9. Interaction among key management modules.

might be better solutions. The ability of capability-restricted nodes in a WAHN to perform public key operations has been questionable due to the scarce computing and power resources. Different symmetric key-based authentication algorithms have been proposed in the literature. Examples include TESLA [30], LEAP [41]. Advances in hardware implementation and power management are expected to overcome this issue Some recent research work investigated the possibility of implementing public key operations in hardware with very low power consumption [19,35].

The ALM module is responsible for maintaining application-level multicast structurle and using this structure to emulate multicast via the unicast at the network layer. Any existing ALM scheme (e.g., [18]) can be used to implement this module. In our prototype, we use Multicast Content Addressable Networks (MCAN] [32], a highly scalable lightweight ALM protocol designed for large population multicast networks. It was shown that the proposed architecture is implementable under current technology. In the coming sections, we discuss the design issues of different components of the distributed group key management scheme, namely, key generation, key distribution, and key assignment.

3.2. COMBINATORIAL KEY DISTRIBUTION SCHEME (CKDS)

In order to provide efficient group communications in wireless ad-hoc networks, we propose Combinatorial Key Distribution Scheme (CKDS). Our scheme depends on EBS and CAN. CKDS targets wireless ad-hoc networks of hundreds to thousands of nodes. We consider a set of N autonomous nodes that participate, or intend to participate, in a secure multicast group with a single source. Nodes communicate via a multi-hop wireless facility. We assume a centralized group controller, *GC*, to be responsible for key generation and construction of rekeying messages. GC may reside at one of the autonomous nodes possibility with additional computing and communication capabilities. The terms nodes and users are used synonymously to refer to multicast group members. In the following sub-section, we provide an overview of CKDS. Then, we present the details of our node partition scheme. Next, we describe two schemes for key distribution, *m*-dimensional multicast and 2D multicast.

3.2.1. CKDS OVERVIEW

In secure group multicast, the main challenge is to change and redistribute keys (rekey) whenever a member joins (or leaves) a group in order to maintain backward (or forward) secrecy. CKDS deals with efficient distribution of the new keys, also known as altered keys in this chapter. In particular, it delivers the *k*-altered keys (based on EBS) to their appropriate group members through hop-by-hop unicast. We keep in mind that efficiency and scalability are major concerns of such a scheme. CKDS follows the following procedure:

1. Assign a *key string* to each node.
2. Construct a multi-dimensional key space, and partition the nodes in that space according to their key strings.
3. Project the key space into m-dimensions corresponding to the keys unknown to the evicted/added
4. Use one of two key distribution schemes, *m*-dimensional multicast, or 2D multicast to convey messages to individual nodes.

Revisiting the EBS example in Table 1 (section 2.6.1), we consider for example, that user U_1 leaves the group. Three keys, K_1, K_2, and K_3, need to be altered to maintain forward secrecy. The group controller generates three new keys K_1', K_2', and K_3' to replace the old keys. The three new keys are encrypted using both K_4 and K_5, generating two rekeying messages R_4 and R_5.

Consider the 10-user case as in Table 1 (section 2.6.1). Applying step 1, we assign each node a bit string similar to the corresponding column in the EBS canonical matrix. For example, node U_1 is assigned "11100." The nodes are then partitioned into a 5-dimensional space based on step 2. Since it is complex to indicate the 5-dimensional figure, we show its projection on the K_3 –K_4 plane in Figure 10. For example, since U_1 contains K_3 but does not contain K_4, it is shown in the 2nd half of the K_3-axis and 1st half of K_4-axis.

Now, we look at step 3. Assuming that node U_1 is leaving the multicast group, according to step 3, the key space is projected based on K_4 and K_5 (the keys unknown to U_1). Applying the key space projection in step 4 to U_1 is shown in Figure 10. It may be

observed that U_1 is in the 1st half of K_4-axis and 1st half of K_5-axis indicating that U_1 has neither of these keys.

Finally, applying step 4 results in distributing the messages according to Figures 10.a and 10.b, for m-dimensional multicast and 2D multicast, respectively. As shown later, to perform rekeying after user U_1 left the space, we use only 12 unicast messages each of size 3 keys for *m*-dimensional multicast. In 2D multicast, we use only 15 smaller messages each with size 1 key and 3 messages of size 3 keys each. In the following sub-sections, we describe the above procedure in more details.

3.2.2. KEY STRING ASSIGNMENT (STEP 1) AND NODE PARTITIONING (STEP 2)

To partition nodes, we start by assigning to each node a bit string of length $k+m$ that we refer to as the *key string*. The key string has bit positions corresponding to the ordered set of $k+m$ keys. A bit value of 1 in a certain position i ($1 \le i \le k+m$) denotes that the node knows K_i, while a bit value of *0* denotes that the node does not know K_i. In EBS

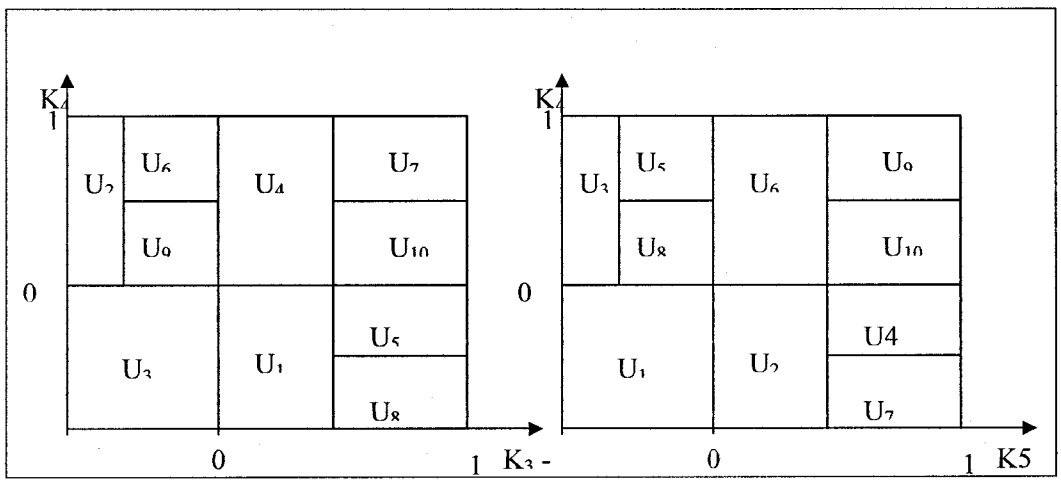

Fig. 10.a. Projection of space on K_3-K_4 plane (ES_3).

Fig. 10.b. Projection of space on K_4-K_5 plane (ES_1).

terminology, key strings are the columns of the EBS canonical matrix A (see Table 1). In CKDS, we arrange nodes in a virtual Cartesian space similar to the CAN virtual space, but with a few differences. Our space has $k+m$ dimensions with one dimension assigned to each of the EBS $k+m$ keys. Each dimension extends between the values of -1 and 1. Each node occupies a portion of this space by having a set of $k+m$ intervals, one for each dimension. An axis splits its dimension into two halves. The negative half has all zones of nodes that do not know that specific key corresponding to this dimension (i.e., those with a key string with a zero at that specific position). The positive half contains the nodes that know the corresponding key. For example, in Figure 10.a , users U_2, U_3, U_6, and U_9 are in the negative half of K_3. This is also evident from the K_3 row in Table 1. The remaining six users are in the positive half.

We refer to the portion of the space that is specified via a bit string of length d as a *length d-quadrant.* A quadrant that satisfies the condition of having k ones in the bit string is considered valid. Every length $k+m$-quadrant can carry a maximum of a single node. Sub-spaces are projection of the key space in a number of dimensions less than $k+m$. Subspaces are divided into smaller-length quadrants. A length *q-quadrants*, where $q \leq k+m$, is considered valid if and only if its bit string contains i ones where $i \leq k$. Our key

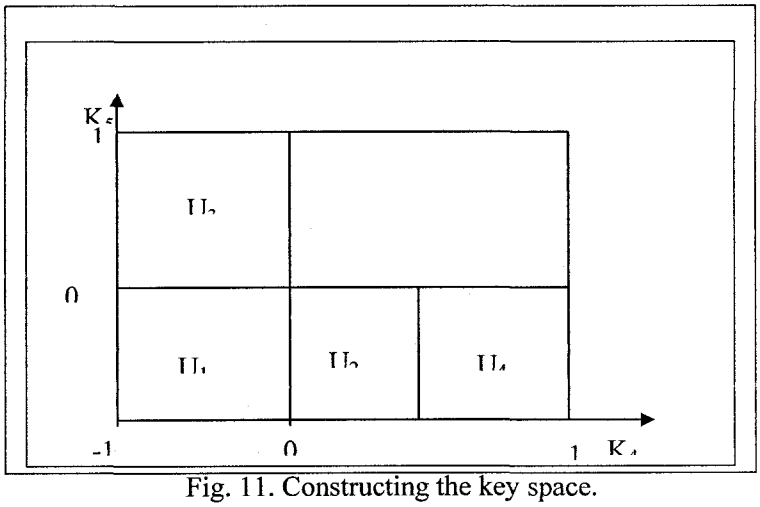

Fig. 11. Constructing the key space.

space construction goes as follows:

- Assume an initial *(k+m)*-dimension empty space divided into positive and negative halves along each dimension to form 2^{k+m} quadrants. We consider only valid quadrants.
- A joining node will be assigned to the quadrant of the space that corresponds to its key string.
- A new node joins the space by contacting an already existing node and reaching the closest node to its key string. (Usually by using Cartesian distance)
- Whenever a node joins a quadrant occupied by one or more existing nodes, one of the existing nodes splits its interval with the new node along the dimension corresponding to the key with lower number.
- Neighbors are updated according to the same rule as in CAN.

The main difference between CKDS key space and the regular CAN space is that each node holds two values along each dimension to represent the interval it is occupying. The sign of such values has significance with respect to known and unknown keys. Clearly, no node can have an interval along a dimension that contains zero. In the following, we illustrate how the CDKS key space is managed as users join and leave.

Consider the EBS(10,3,2) example. We only consider the K_4K_5-plane for simplicity. The first user U_1 joins the space with *0* in the fourth and fifth bit of its key string. The user is assigned the entire *00*-quadrant and occupies one quarter of the available K_4K_5-plane. We notice that user U_1 has no neighbors it is the only one in the space. The next user U_2 joins with k_4-k_5 string *10* (meaning that he or she has K_4 but not K_5). According to the same rule, U_2 is assigned the *10*-quadrant and occupies another quarter of the plane. The same scheme will place user U_3 in the *01*-quadrant. The next user, U_4, has the key string *10*, same as user U_2. This will force U_2 to share its space with U_4, by dividing the space in half along the K_4 dimension. This situation is illustrated in Figure 11. The next user U_5 will share the *01*-quadrant with U_3. U_6 will be located in the *11*-quadrant. U_7 will have to share the *10*-quadrant with both U_2 and U_4. This will force one of these two users to split its space (this time along K_5) and share it with U_7. The same idea will be applied to accommodate U_8 in the *01*-quadrant. Users U_9 and U_{10} will have to share the *11*-quadrant with U_6 in the same manner. The final K_4-K_5 plane will look as in Figure 12.

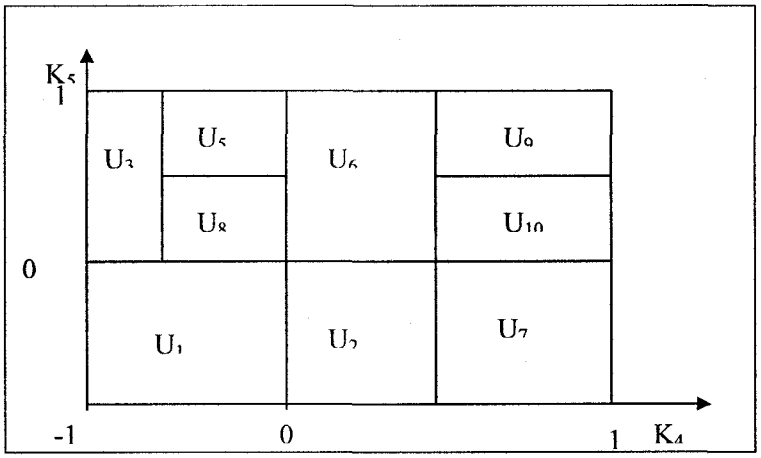

Fig. 12. Projection of space on K_4-K_5 plane after U4 left.

Each user in the above illustration occupies some area of the K_4-K_5 plane and recognizes its neighbors along both dimensions. For example, U_8 recognizes U_3 and U_6 as his preceding and succeeding neighbors along K_4. At the same time U_{10} recognizes U_9 and U_4 as his preceding and succeeding neighbors along K_5. In general, in a perfectly partitioned *d*-dimensional space, each user will keep track of *2d* neighbors 2 of them along each of the *d* dimensions.

An important difference between this organization and the one in CAN [32] is that users do not recognize neighbors with whom they do not share key(s) along the specified dimension(s). In other words, neighborhood does not the axis. In the above example, U_8 does not recognize U_1 as its neighbor along K_5 although they overlap along the K_5 dimension. This property makes the *all-zeros*-quadrant unreachable from other quadrants. However, users located in the *all-zeros*-quadrant in a certain 2D plane will be occupying other quadrants in different 2D projections. For example, in Figure 12, U_1 and U_8 are not recognized as neighbors along K_5, while in the K_3-K_4 plane in Figure 10.a, they are recognized as neighbors along K_3.

A user might leave the key space either voluntarily or involuntarily. In both cases the same scenario applies. All neighboring nodes will be able to sense the absence of the leaving node. However, the neighbors within the same quadrant take responsibility to regain the space that had been occupied by the absent node similar to CAN.

In the above example, assume that U_4 leave the key space (may be evicted or went down). Users U_2, U_{10}, and U_7 will recognize the absence of U_4. As U_7 is the node with which U_4 shared the space, U_7 will regain the space and update the surrounding neighbors with the new state. The new space is illustrated Fig. 12. Key Distribution in CKDS

Rekeying usually involves three steps key generation, construction of rekeying messages, and key distribution, as described in Section 1. Key distribution involves delivering the rekeying messages to appropriate group members. In a centralized key management system, the group controller is normally responsible for these three steps. In EKDS, the group controller is only responsible for the first two tasks. The step 3 of key distribution is delegated to the nodes themselves using our key distribution scheme.

In fixed networks with network-level multicast support, key distribution is not an issue as the group controller just multicasts the rekeying messages on the channel to everyone. With ad-hoc networks using application-level multicast, key distribution is not as simple. Since the multicast is implemented with an underlying unicast, or at best as a very limited multicast in some areas, the number of unicast messages needs to be optimized.

We start by revisiting the EBS(10,3,2) example with our two rekeying messages R_4 and R_5. A first-cut scheme is to use Directed Flooding [32] multicast algorithm as described in Section 2 to distribute the keys. This scheme might not achieve acceptable performance due to the unnecessarily large number of unicast duplicate messages used to deliver updated keys to users.

Here, we propose two different unicast distribution schemes-*m-Dimensional* scheme, and *Localized 2D multicast* scheme. Before going into the details of our two schemes, we need to define the terms *Exclusion sub-space*, and *Complementary nodes*.

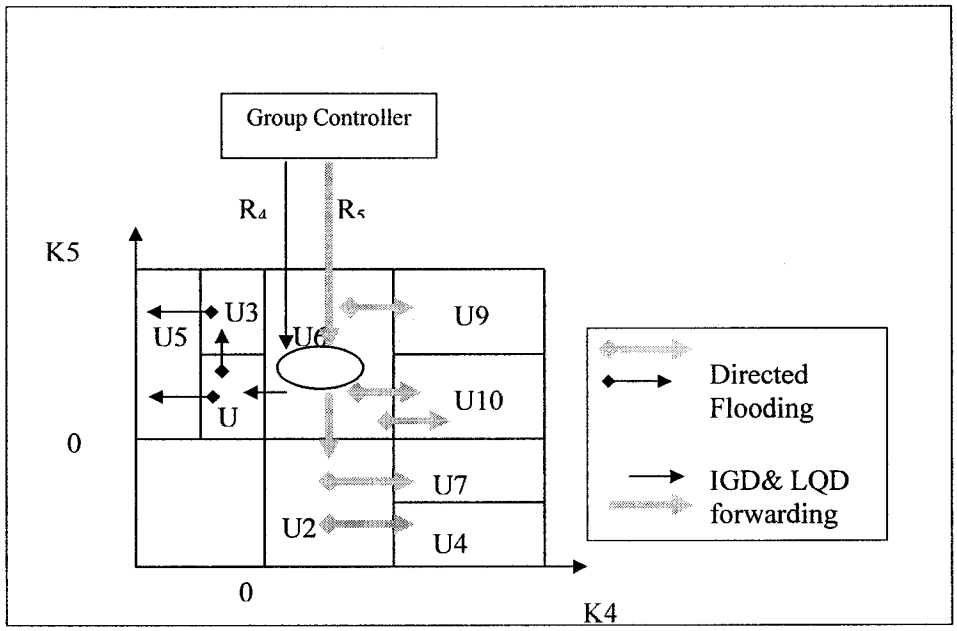

Fig. 13. m-Dimensional key distribution.

An Exclusion sub-space (ES_i) is defined for a group member U_i as the m-dimensional subspace representing the projection of the group ($k+m$) dimensional key space along the m dimensions whose keys are unknown to user i. Figure 10.a and 10.b represent ES_1 and ES_3 respectively. Exclusion sub-spaces represent the projection of the key space in a way that a certain user looks isolated in an *all-zeros*-quadrant. The main purpose of this projection is to facilitate rekeying in case a user leaves or joins. In the exclusion sub-spaces, we define quadrant boundaries as virtual lines (or hyper-planes) that separate quadrants. In the following context, unless stated otherwise, rekeying messages are blocked by such boundaries.

A complementary node for a user U_i is a node that knows all keys that are unknown to U_i. In the above example, nodes U_6, U_9, and U_{10} are complementary to U_1. For a certain user U_i, all the complementary nodes are located in the *all-ones* quadrant in ES_i.

3.2.2.1. THE M-DIMENSIONAL MULTICAST SCHEME

In this scheme, the group controller delegates the responsibility of key distribution to some other nodes in the key space. For adding or evicting a certain user U_i, the group controller projects the key space into $ES_{i,,}$ the exclusion sub-space of this specific user.. The key generator tries first to locate any node of the complementary set of U_i. A complementary node knows the entire set of the *m*-unaltered keys. If such a node exists, the key generator forwards the rekeying messages to that node. The most suitable node is the one occupying the lower left corner of the *all-ones*-quadrant. Such a node will be able to forward rekeying messages to nodes in other quadrants (clearly, except the *00* quadrant), it forwards rekeying messages to other nodes with which it shares the *all-ones*-quadrant at the same time. We refer to such a node as the *Initial Global Distributor (IGD).* If such a node does not exist, the group controller plays this role of IGD. The group controller acts as a hyper-node with an all-ones key string, thus it can play the role of IGD.

The initial distributor has shared boundaries with all nodes eligible to receive the rekeying messages. It first locates a central node in each quadrant and forwards the appropriate rekeying message to that node. We refer to such a central node as *Local Quadrant Distributor (LQD).* A central node is selected as LQD to minimize the delay. LQD is responsible for multicasting the received rekeying message among all other nodes at the same quadrant using a multicast algorithm such as Directed Flooding. The GID plays the role of LQD within its own quadrant (i.e. the *all-ones*-quadrant).

Nodes at any specific quadrant may know more than one key out of the m keys used by the group controller for encryption. This situation may result in a large number of duplicates since all nodes in such a quadrant will receive different encryptions of the same messages. To avoid such an overhead, the group controller forwards only one message to each quadrant. One idea is to send the message encrypted with the key of minimum (or maximum) index that nodes in a certain quadrant know.

In Fig. 13, we apply this scheme to our EBS(10,3,2) example. The group controller selects U_6 as the IGD. U_6 will select U_2 and U_8 as LQD for the *10*-quadrant and *01*-quadrant and forwards to them R_4 and R_5 respectively. As a LQD, U_6 distributes the rekeying message R_5 to the *11*-quadrant.

3.2.2.2. LOCALIZED 2D MULTICAST

In the above approach, rekeying is done through encrypting all updated *k* keys using their corresponding older revoked keys, and then encrypting the whole message once with very one of the *m* unaltered keys. In such an approach, two cascaded encryptions are done for each rekeying message. Some users get longer messages where they need only small part of it. However, they can't trim this part until they decrypt the message to unleash the outer encryption. This overhead might be significant in a wireless ad-hoc network with limited networking resources and processing power at each node. In order to reduce this traffic and processing overhead incurred by individual nodes, we propose 2D-multicast, an alternative key distribution scheme with lower traffic and decryption overhead.

As an example, consider the EBS(10,3,2) above. The rekeying message R_4 contains K_1, K_2, and K_3. This message will reach three users, U_3, U_5, and U_8. U_3 needs to update only K_1 and K_2, U_5 needs to update only K_1 and K_3, and U_8 needs to update only K_2 and K_3. Each of these users receives an extra key in the rekeying message. The situation is even worse in the *11*-quadrant where each user receives 2 extra keys in the rekeying message. This constitutes an overhead of one or two thirds of the total traffic. In an environment with low resources, such as wireless ad-hoc networks, such an overhead is considered significant.

One idea is to break the rekeying message down into smaller messages, say *m* disjoint messages each of which carries only one key. Thus,

$$R_4=K_4(K_1(K_1'), K_2(K_2'), K_3(K_3')),$$

will be broken into 3 messages as follows.

$$R_{41}= K_4(K_1(K_1')), R_{42}= K_4(K_2(K_2')), R_{43}= K_4(K_2(K_3'))$$

Each of these messages will be directed to the specific set of users who need such a key. We refer to the key used for outer encryption (such as K_4 in R_4) as the *encryptor* key. The key whose new version is encrypted using its old version within the message is referred to as the *altered key.* In order to avoid double decryptions by each user, we can use a new key for encryption. This new combined key is constructed by using a one-way hash function to hash the altered key with the encryptor key. According to the security requirement of the network and the computational capabilities of the individual nodes,

the hash function can be as simple as a bit-wise Exclusive Or, or as complex and secure as SHA-1. We denote such combined key as $K_i|K_j$, where K_i is the encryptor key and K_j is the altered key. In that case, the original R_4 will be broken down into 3 simpler messages as follows.

$R_{41}= K_4|K_1(K_1')$, $R_{42}= K_4|K_2(K_2')$, $R_{43}= K_4|K_3(K_3')$

In general, a rekeying message to update a certain altered key, K_j encrypted with a certain encryptor key K_i, will be in the form, $R_{ij}=K_i|K_j(K_j')$.

For these new messages to be distributed, we consider a projection of the key space in the two dimensional plane corresponding to keys K_i and K_j. In such projection, we consider only the *11* quadrant, the quadrant in which all nodes have the two keys K_i and K_j. Distribution in such quadrant starts with virtual node that occupy the *11* lower left corner and plays the role of LQD. One of these nodes may even represent the group controller. Messages are multicasted to all nodes in the *11*-quadrant using the same Directed Flooding algorithm.

Using individual messages for each set of users who know two keys will constitute a huge traffic overhead. A better way is to use this message splitting technique within each of the length *m*-quadrants. LQDs in every quadrant are responsible for multicasting the rekeying messages. The group controller forms a single rekeying message for each encryptor key concatenating the individual rekeying messages for all altered keys that has the same encryptor in order. It then sends to each quadrant the rekeying messages encrypted with the encryptor that has maximum (or minimum) index the quadrant nodes know. The distribution node distributes the keys differently as follows.

For each distribution node that receives a rekeying message Ri with encryptor key k_i

For each altered key K_j in R_i

Project the quadrant space in two dimensions Ki-Kj and consider only the 11-quadrant.

Distribute the part Ki|Kj(Kj') to the 11 quadrant using directed flooding

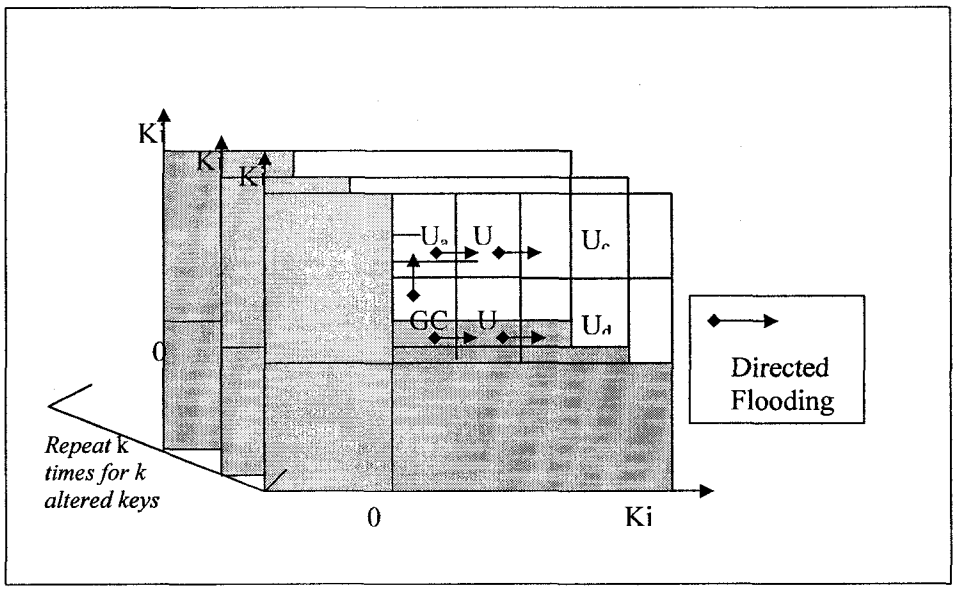

Fig. 14. Localized quadrant multicast with 2D rekeying.

According to this scheme, in a particular length-m quadrant, multiple 2D projections are used to distribute the rekeying messages. Specifically, one projection for each pair of keys K_i and K_j that at least known but a single node in the quadrant. Fig. 14 depicts this multiple 2D distribution. Each node receives only the relevant keys without any overhead traffic. The tradeoff here is between traffic overhead and the number of unicast messages. This is expected to be relevant in certain environments where nodes have even more limited resources such as sensor networks. The overhead incurred by each node in receiving irrelevant keys may achieve better overall performance with lower number of messages in some other environments.

3.3. THRESHOLD-BASED KEY GENERATION IN WAHNS (TKGS)

We assume a WAHN of hundreds to thousands of nodes. A set of *N* autonomous nodes participates, or intends to participate, in a secure multicast group with a single

source. Nodes communicate via a multi-hop wireless facility and are prone to failure or mal-behaving. No centralized key generator is assumed.

In order to provide dependable and efficient secure group communication service in such environment, we start with an EBS-based key assignment. Key distribution is performed via any of the two CKDS key distribution schemes presented in Section 3.2. We propose a collaborative verifiable threshold-based key generation scheme to replace the centralized group controller. Fig. 15 shows a proposed general architecture for secure group communications in WAHNs. In this architecture, no network-level support for multicast is assumed. A group manager module performs necessary group key generation and re-keying initialization. A key distribution module, such as CKDS, performs key distribution based on unicast multi-hop message service. In Fig. 1616, we show some more details about the proposed group manager that uses our proposed TKGS.

3.3.1. THRESHOLD-BASED KEY GENERATION SCHEME (TKGS)

In order to design our secure threshold key generation scheme, we propose a simple key generation mechanism. For each administrative key K_i, $1 \leq i \leq k+m$, in the original EBS, we define a key generation key, KGK_i. KGK_i is used to generate $K`_i$, the updated version of K_i, through a simple cryptographic function C. Thus,

$K_i`=C(KGK_i,K_i)$

3.3.1. THRESHOLD-BASED KEY GENERATION SCHEME (TKGS)

In order to design our secure threshold key generation scheme, we propose a simple key generation mechanism. For each administrative key K_i, $1 \leq i \leq k+m$, in the original EBS, we define a key generation key, KGK_i. KGK_i is used to generate $K`_i$, the updated version of K_i, through a simple cryptographic function C. Thus,

$K_i`=C(KGK_i,K_i)$

To make it more systematic, we consider all the administrative keys to depend on discreet intervals. At time t_j, we have the key K_i takes on the value $K_{i,j}$, instead of using the notation of $K_i`$. Accordingly,

$K_{i,j}`=C(KGK_i,K_{i,j-1})$

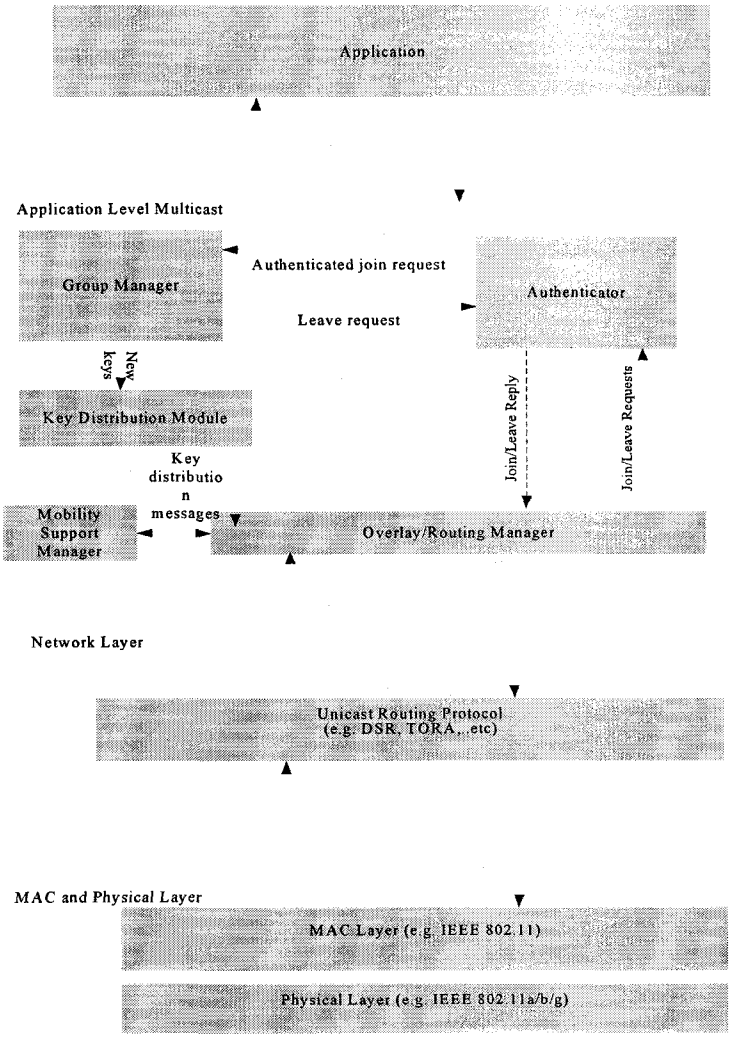

Fig. 15. Architecture for secure group communications in WAHNs.

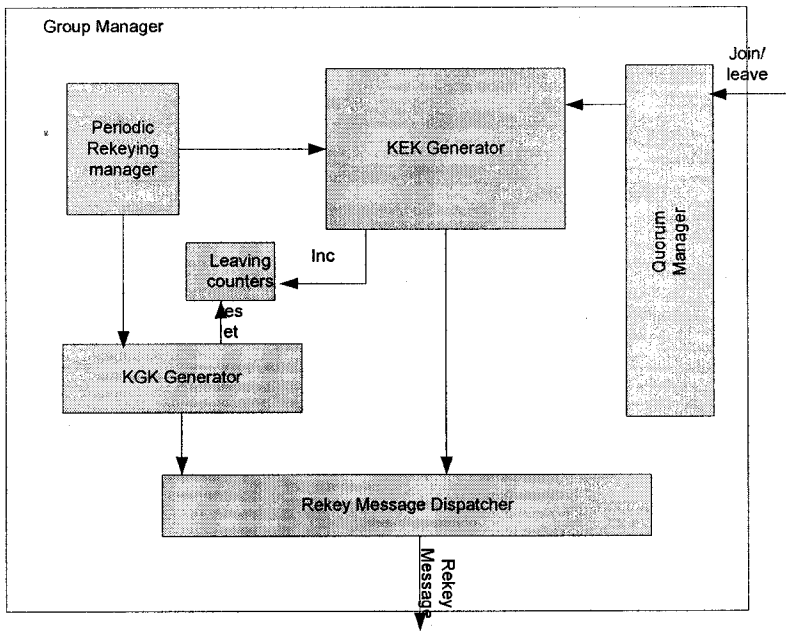

Fig. 16. Architecture of the Group Manager.

The cryptographic function C should be secure and should be known to all nodes with relatively simple computation. The KGKs are not stored anywhere via participating nodes. They are used as the shared secret for the VSS scheme. The secret is shared among all nodes that know the administrative key K_i. To apply a *t-out-of-N* VSS [42] scheme, we assign N shares, one to every node in the set of nodes knowing K_i. According to CKDS [80], this set contains $N_i = \binom{m+k-1}{m}$ nodes. Any t nodes out of n suffice to reconstruct the KGK and generate the new updated key.

As illustrated in Fig. 16, rekeying can be done either periodically or as needed. Although subjecting the system to short vulnerability windows, periodic batch rekeying was shown to be more efficient in hierarchical rekeying structures such as LKH [118]. Periodic batch rekeying improves efficiency by reducing number of rekey messages to be signed as well as taking advantage of join and leave overlap.

In our proposed threshold-based rekeying scheme, rekeying is performed on individual administrative keys using a *t-out-of-N* threshold scheme. The set of nodes that know a specific administrative key K_i represent the community $\{N_i\}$, while any t nodes out of this community can form the quorum for rekeying. In order to perform a verifiable sharing of KGKs, we use a technique similar to NEW-VSS [42]. To perform rekeying of K_i, we assume a secret sharing polynomial $s_i(x)$as well as a verification polynomial $v_i(x)$, both of degree *t-1*, for each key generation key KGK_i corresponding to an administrative key K_i, where t is the specified threshold. Each party, U_j, holds a share of both polynomials, denoted as, $s_i(j)$ and $v_i(j)$, as well as a commitment function *C*. We outline *TKGS*, our simple key generation protocol in Fig. 17.

The above basic TKGS provides systematic collaborative key generation in an environment with un-trusted nodes. TKGS tolerates faulty and misbehaving nodes as long as there could be up to t valid participants. The key generation can take place only if

TKGS (k_i)

1. *A group member, referred to as the dealer $d_{i,t}$, broadcasts a request for rekeying of K_i to other members of $\{N_i\}$.*
2. *Each user, U_j in the community $\{N_i\}$ broadcasts his own response to the rekeying request according to a certain probability.*
3. *Upon receiving at least q_{rekey} positive responses to the request, the dealer $d_{i,t}$ assumes the request was granted and broadcasts a ready-to-accept-shares message.*
4. *Each user, U_j in the community $\{N_i\}$ may send its share of generation key, $KGK_{ij} = s_i(j)$ as well as its share of the verification function $v_i(j)$, to the dealer via unicast.*
5. *Collecting at least t confirmation messages with verified shares, the dealer may proceed with the rekeying, or time out*
6. *The dealer reconstructs the polynomials $\hat{}s_i(x)$ and $\hat{}v_i(x)$ using t valid points of verified shares. Then, the dealer verifies the reconstructed polynomials using N_i values of the commitment function C.*
7. *If the verification fails, the dealer broadcast a complaint message and a different node restarts the procedure.*
8. *If the verification succeeds, the dealer reconstructs KGK_i and generates the administrative key $K_{i,t+1}=C(KGK_i, K_{i,t})$*
9. *The key distribution procedures takes place as in CKDS*

Fig. 17. Threshold Key Generation Scheme, TKGS.

a quorum of at least q_{rekey} vote to rekey. The parameter t can be tuned to reflect the amount of trust in the environment. In case most nodes are un-trusted, we may use a larger value of the threshold t at a higher communication and computational overhead. On the other hand, trust worthier environment can use a smaller threshold with lower overhead.

A simple DoS attack on the basic TKGS may take place by having multiple nodes place unnecessary rekeying requests and cause system thrashing. The parameter q_{rekey} reflects the amount of trust placed on a specific individual node to play the role of the dealer and initiate rekeying. Higher value of q_{rekey} means that more nodes should approve the act of rekeying before it takes place and vice versa. Since nodes multicast their responses, different participant can verify the approval of rekeying before sending their shares. For secure transmission of shares, shares are sent encrypted with a symmetric key derived from all keys shared between the shareholder and the dealer.

3.3.2. TKGS/V

Although the basic TKGS protects the key generation process from faulty and mal-behaving nodes, a dealer might falsify the generated key. Since no other node has the whole t verified shares, the dealer might correctly collect the shares and then distribute a different key. We suggest two techniques to protect the key generation from mal-behaving dealer. The first technique is through replication, i.e. replicate the key generation process with more than one dealer and compare the generated keys for verification. The second technique is through backward confirmation, i.e. nodes that played the role of dealer through a certain past window of time are responsible for confirming the key generation process and comparing it with the received key. Those nodes can multicast a complaint to all nodes in order to nullify the rekeying process.

Replication is done simultaneously at all participating dealers and the keys are multicast to all members in $\{N_i\}$. It is the responsibility of individual nodes to verify the received keys. Clearly, using replication incurs an overhead of repeating the *Basic TKGS* as many times as the number of replicas. On the other hand, in backward confirmation, verification is done after the key distribution. Since all past dealers have the verified share, they can use the current key K_i and the shares to replicate the process done by the current dealer. Accordingly, no communication overhead is incurred on other nodes. This

TKGS/V (k_i)

Dealers

1. *Start with (Rekeying Number) RN=0*
2. *for ($j= 0; j \leq q_{dealer} - RN_r ; j++$)*
   a. *Select dealer d_j according to select criteria*
   b. *Dealer d_j initiates Basic TKGS(k_i)*
   c. *Dealer d_j distributes K_i*
3. *RN++*
4. *For each past dealer d_k, $1 \leq k < RN$*
   Apply steps 6, and 7 of Basic TGKS

Members

1. *Upon approving rekeying by each dealer d_k, separately, $1 \leq k \leq q_{dealer} - RN$ send your shares as in step 4 in Basic TKGS.*
2. *Compare K_i received from each dealer, and verify it is the same*
3. *If you receive complaints from a quorum of past dealers, you may ignore the rekeying, and restart*

Fig. 18. Threshold Key Generation Scheme with Verified dealer, TKGS/V.

might not be suitable for systems with tight time constraints. Another restriction is the availability of past dealers especially at system startup.

In TKGS/V, we propose the use of both replication and backward confirmation. We define a dealer quorum parameter q_{dealer}, to determine which technique is used. At the system start up, and as long as number of rekeying time, and accordingly number of past dealers, less than q_{dealer}, we use replication. When the number rekeying times exceeds q_{dealer}, backward confirmation is used. Fig. 18 depicts TKGS/V.

3.3.3. CHANGING KGKS

The above TKGS/V provides verifiable key generation as long as KGK_i does not change. Every time rekeying of K_i is performed, a new node plays the dealer role and reveals KGK_i. If a large number of nodes reveal KGK_i, KGK_i cannot be considered secret anymore. At the same time, if a node that had played as a dealer left the group, then KGK_i, and accordingly, all future $K_{i,t}$ are revealed. We consider each node that has played the role of dealer, since the last KGK change, as a type 2 node. Theoretically, any group dynamics, i.e. join or leave, related to type 2 nodes require *KGK* to be changed. We assume that joining nodes are always of type 1, knowing only k administrative keys. When a node is selected to play the role of dealer for a certain key generation of any of the administrative keys known to it, the node is promoted to be a type 2 node.

Altering the key generation key, KGK_i, is an expensive operation similar to the original system bootstrapping. This process maps to the sharing phase of the original VSS protocol. We assume *KGKs* to change less frequently than administrative keys. In order to ensure less frequent updates, we may place some policy on selecting a specific node to act as a dealer. Some parameters to consider in this situation is the probability of that specific node to leave the group, number of times the node has acted as a dealer for the specific key K_i as well as for other administrative keys known to it, the specific node failure probability if available, etc.

To handle the above situation, and to keep backward secrecy at the type 2 nodes, we need to change KGK_i not only upon the leaving of any type 2 node with respect to KGK_i, but also when the number of previous dealers of the corresponding administrative key K_i reaches a certain threshold, say 30% of the total population, N_i. Rekeying *KGKs* will hamper the collusion of past dealers as well as mobile adversary attacks. Assuming a mobile adversary is capable of compromising up to $c_{compromized}$, say 2%, of N_i at a certain

window of time, T<t, measured in number of K_i rekeys. We can set the threshold at which, we change KGK_i, t_{KGK} to be less than $t/c_{compromized}$-1, so that before the mobile adversary achieves the required t compromised nodes, the corresponding KGK_i changes.

3.4. KEY ASSIGNMENT AND MITIGATING COLLUSION (CKAS)

In the remaining part of this chapter, we focus on Combinatorial Key Assignment Scheme (CKAS), our solution to the key assignment and batch issues in EDKMS. The main objective of CKAS is to assign keys to group member in a way that facilitates and minimizes rekeying. In CKAS, batch rekeying is used instead of individual rekeying (after each user joins or leaves) to improve system performance. In addition to batch rekeying, users are sub-grouped into clusters either based on physical location and/or behavioral characteristics (such as expected departure time). In order to support such sub-grouping, we extend the original EBS scheme into a Clustered Exclusion Basis System (CEBS). Finally, we propose a simple key assignment scheme in which, keys are assigned to users in a way that reduces the need for rekeying.

3.4.1. BATCH REKEYING AND COLLUSION

Although subjecting the system to short vulnerability windows, periodic batch rekeying was shown to be more efficient in hierarchical rekeying structures such as LKH [114]. Periodic batch rekeying improves efficiency by reducing number of rekey messages to be signed as well as taking advantage of join and leave overlap.

In EBS system, batch rekeying involves even more system vulnerability due to possibility of collusion. Evicting a number of users, c, without rekeying allows those users to exchange system keys and to be able to beat the system. On the other hand, batch rekeying improves efficiency significantly specially in high dynamic groups. The main question in batch rekeying is usually when to rekey. In case of joins, since joining users do not know any system keys as long as no rekeying has happened, rekeying is either periodically or upon reaching a certain number of joining users. A study of the effect of rekeying period on the performance can be found in [118]. In case of leaving users, the number of leaving users and the keys they know induces a certain chance of collusion that can beat the system.

In EBS, collusion of c users happens when those users collectively know the whole set of k+m EBS administrative keys. In clustered EBS, collusion happens when a group of c evicted users collectively know the entire set of $k_c+m_c+k_g+m_g$ administrative keys. In this chapter, we consider only total collusion rather than partial collusion. Collusion is the main obstacle for using batch rekeying in EBS-base key management. Rekeying should take place before collusion can occur. In case collusion occurs, the only recovery can be through individual user rekeying in a way similar to GKMP [31], which incorporates O(N) messages.

In CKAS, users are allowed to leave and rekeying is performed on a batch basis. Since they might exchange key information before rekeying takes place, leaving users in the rekeying period are considered potential colluders. Clearly, the more potential colluders are left in the system the higher the collusion probability is. Batch rekeying takes place upon reaching a certain collusion probability to avoid expensive collusion recovery. We present a model for collusion probability evaluation. Rekeying takes place as the system reaches a threshold collusion probability and accordingly a corresponding threshold number of evicted users. If the number of potential colluders is lower than a certain value (c_{min}), the probability of collusion is zero, and no rekeying is considered. After the number of colluders reaches a certain value (c_{max}), collusion is certain since the potential colluders already have all system keys. For all values in-between, the collusion probability should be evaluated and based on that rekeying might take place.

3.4.2. CLUSTERED EXCLUSION BASIS SYSTEM (CEBS)

In many group applications, user behavior (e.g. joining and leaving the group) is not purely random. Users who join the group simultaneously may leave also simultaneously. Due to limited communication capabilities in a WAHN environment, the group might be participating in several clusters according to their locations. Communications between nodes belonging to two different clusters might be infeasible. An efficient key management schema should take advantage of such property to facilitate rekeying. We introduce CEBS, an extension to the EBS key management scheme with clustering support.

In CEBS, two levels of EBS keys are used to support clustering. If we are using originally $k+m$ keys, of which k keys are known by each user, then we can divide this set as follows. Each user knows k keys, out of which, k_g are global keys (used across clusters) and k_c are local keys (used within the cluster), where $k=k_g+k_c$. The same for the unknown keys, each user have m unknown keys (represented as zeros in his key string), m_g out of which are global and mc are local, where $m=m_g+m_c$.

In such solution, each cluster has the internal structure of EBS(N_c,k_c,m_c) while the outer group of clusters has the structure of EBS(N_g,k_g,m_g). The maximum number of clusters we can have is N_g and the maximum number of users within any cluster id N_c, where

$$N_g = \binom{k_g+m_g}{k_g} \qquad N_c = \binom{k_c+m_c}{k_c}$$

The total number of users supported by the set of $k+m$ keys are $N^*=N_g*N_c$. The minimum number of colluders to reveal all k_c+m_c keys within the cluster is referred to as $c_{min,c}$. Parameters k_c and m_c may be selected to guarantee a minimum number of colluders required to reveal cluster keys.

All users within the same cluster have the same assignment of global keys. Colluding to reveal cluster keys requires a minimum number of colluding clusters, $c_{mn,g}$, represented as at least one individual user from each of the $c_{mn,g}$ clusters. Selecting k_g and m_g to guarantee a reasonable $c_{min,g}$, along with the assumption that inter-cluster communications are infrequent, makes such collusion unlikely.

The formal specification of CEBS, a clustered EBS system of N_g clusters with N_c users/nodes within each cluster, k_g+k_c keys out of $k_g+k_c+m_g+m_c$ keys known by each user CEBS(N_g, N_c k_g, k_c m_g, m_c) can be described as a cascaded organization of two EBS systems, an inter-cluster system of EBS(N_g,k_g,m_g), and an intra-cluster system of EBS(N_c,k_c,m_c).

The number of clusters, N_g should satisfy

$$N_g \leq \binom{k_g+m_g}{k_g} \qquad (1)$$

The number of users per cluster, N_c should satisfy

$$N_c \leq \binom{kc+mc}{kc} \tag{2}$$

From (1) and (2), the total number of users in CEBS(N_g, N_c k_g, k_c m_g, m_c), N should be

$$N = N_c N_g \leq \binom{k_g+m_g}{k_g}\binom{k_c+m_c}{k_c} \tag{3}$$

3.4.3. KEY ASSIGNMENT SCHEME

The way keys are assigned to users in EBS affects the collusion possibility. In this context, key assignment refers to assigning to a specific user a key string that tells which keys are known and which are unknown to him/her. The objective of a key assignment scheme is to assign keys to users expected to be evicted together so that the collusion probability is minimized. If key strings are assigned improperly o co-evicted users, the case of having c_{min} colluders who are able to reveal all keys and accordingly rekeying has to be done more frequently (individually if c_{min}=2). On the other hand, if keys are assigned properly to those potential colluders, rekeying can be deferred till we have almost c_{max} evicted users (e.g. we can use a higher value for the rekeying threshold based on the collusion probability).

The following scheme assigns keys to a set of users in a plain EBS (who are expected to collude) so that the collusion probability is minimized.

Input: EBS(N,k,m) canonical matrix, c joining users, c

Output: c key string assignment for c users

Step 1: give the first user the string $(11..1)_k(0..0)_m$

Step 2: for zeros=m-1 to 0 do

> *Give each one of C(k+m-zeros,k) user one of the key strings with consecutive zeros zeros at the end.*

The above scheme guarantees that users who are assigned key strings at one iteration of the loop get the maximum possible key overlap and accordingly the minimum collusion probability. If $c<c_{max}$, the above assignment is collusion free.

In CEBS, if clustering is logical, the system can fill each cluster with up to c_{max},c users. Additional users can be distributed on other clusters according to the same criterion applied to the cluster EBS structure, EBS(N_g,k_g,m_g).

3.5. CONCLUSION

In this chapter, we introduced our proposed solution for key management in WAHNs. We started with key distribution and proposed using Combinatorial Key Distribution Scheme (CKDS). We followed with key generation using Threshold-based Key Generation in WAHNs (TKGS). For key assignment, we proposed Combinatorial Key Assignment Scheme (CKAS) and Clustered Exclusion Basis System (CEBS)., witch assigns closer key strings to co-located nodes. We claim that our architecture can readily be populated with components to support objectives such as fault tolerance, full-distribution and scalability to mitigate WAHNs constraints. In the next chapter, we present a simulation study that compares our proposed solution to existing ones and shows that it gives better results.

CHAPTER IV

SCHEME RESULTS IN WIRELESS AD-HOC NETWORKS

In this chapter, we review simulation results of the three parts of our proposed scheme for WAHNs. In Section 4.1, we view the simulation results of CKDS. In Section 4.2, we overview the simulation results of TKGS. In Section 4.3, we overview the simulation results of CKAS

4.1. CKDS RESULTS

In key management systems such as [18] [30] [94], the system performance is usually evaluated in terms of the number of key changes or the number of multicast messages. Tree-based systems such as the LKH model achieve a logarithmic number of rekeying messages in terms of the number of users [18], [94]. Particularly, in classical LKH, every changed key is multicast once for every descendant of the node holding the key. Some researchers achieved a better performance by using hybrid approaches [3, 42], or by optimizing the tree [133],[82].

In a general EBS(N,k,m), rekeying involves altering k keys and generating m rekeying messages. The relationship between the number of users N and the parameters k and m can be determined by (4).

$$\binom{k+m}{k} \geq N \qquad (4)$$

The number of keys changed per rekeying is at most ($\log_2 N + \log_2(\log_2 N)$), which is less than $2\log_2 N$, the number of key changes in LKH systems such as in [18].

The assumption of a reliable multicast channel results in a direct proportion relation between the number of keys changed and number of multicast messages. In an application-level multicast, this assumption is not valid anymore, since all messages that used to be multicast are now delivered through unicast. In our performance study, we focus on the number of unicast messages as a performance parameter of the network

overhead. Since we target wireless environments with limited resources, we expect nodes to have limited computational power. We consider the average number of decryptions performed by a single node as a performance parameter that represents the computational overhead incurred at individual nodes.

4.1.1. A BASE-LINE CASE

To start with a base-line case, we consider an average system of N=10,000 users supported through EBS. EBS offers a tradeoff between the number of keys k known by each user, and the number of rekeying messages (m). In Fig. 19, we plot the number of keys per user, k, against number of messages, m, for N=10,000 users. There are 7 different (k,m) combinations to support 10,000 users. Our basic comparison case will consider the simplest scheme of distributing m rekeying messages to all users using the directed flooding as in Section 2. In such a case, we have m rekeying messages to be delivered to at most $\binom{k+m}{k}$ users using directed flooding on an m-dimensional space. If

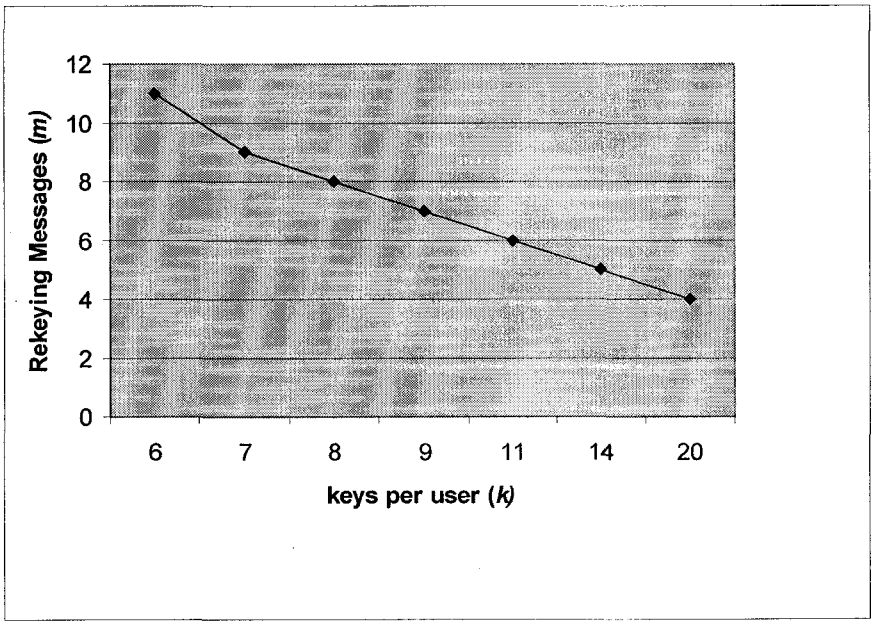

Fig. 19. Rekeying messages Vs Number of Keys per users for 10,000 users.

we assume a perfectly partitioned space, we will have a single message per user. In such a case there are $\binom{k+m}{k}$ -1 unicast messages per rekeying message resulting in a total of m *($\binom{k+m}{k}$ -1) messages, each of size k.

Another option is to use GKMP [49]. In such a case, the group controller will communicate the TEK individually to each group member using her personal key. The number of rekeying messages is clearly N. However, if we assume a uniform two-dimensional distribution of the nodes in the space, each message needs to travel in average along a number of hops equal to the $\frac{1}{2}\sqrt{N}$ to reach its destination. This forwarding introduces a large overhead of unicast messages. Fig 20 shows the number of unicast messages for 10,000 users using both Directed Flooding and GKMP under different (m,k) pairs.

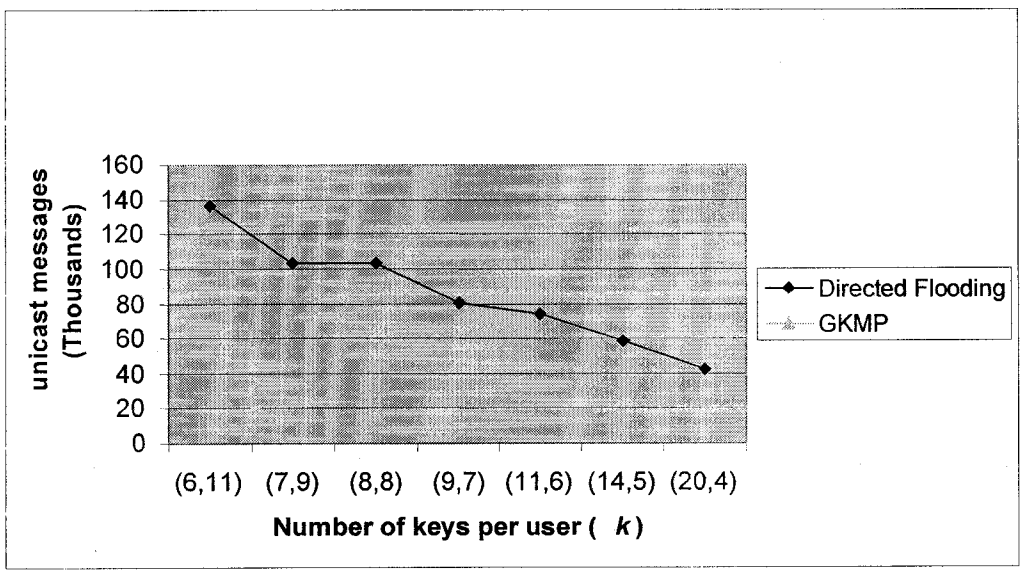

Fig. 20. Number of unicast messages for 1000 Users
Under Directed Flooding and GKMP.

In the simple directed flooding approach, each user performs a number of decryptions upon receiving a certain rekeying message, denoted by N_d. A specific user can share at most *k-1* keys with the evicted user, thus she may perform a maximum of *k-1* decryptions for her own keys plus an extra decryption for the outer message encryptor. Generally, if a user U_i knows *i* keys out of *m*-unaltered keys, this user will perform *k-i+1* decryptions. U_i should know a total of *k* keys out of which, *i* keys are among the *m*-unaltered keys and *k-i* keys are selected from the remaining *k* keys. For a specific value of *i,* the number of users who satisfy the above condition is $\binom{m}{i}\binom{k}{k-i}$. Clearly, A user can know a maximum of *m* keys out of the *m*-unaltered keys. At the same time, a single user cannot know more than *k* keys. Thus, the maximum possible value of *i* is either *k* or *m* whichever is lower. Accordingly, the average number of decryptions done by each user, N_d for N users in the system, is determined by (5).

$$N_d=\frac{1}{N}\sum_{i=1}^{\min(k,m)}\binom{m}{i}\binom{k}{i}*(k-i+1)=\frac{k^2+m+k}{N(m+k}\binom{m+k}{k} \qquad (5)$$

The proof of the N_d final value can be found in Appendix B.

In GKMP, each user performs only one decryption upon receiving each rekeying message, i.e. N_d=1. This lower N_d is achieved at the expense of done linear number of key change per rekeying. Figure 21 lot the above N_d for directed Multicast with 10,000 users under simple Directed Flooding.

4.1.2. ANALYZING M-DIMENSIONAL MULTICAST

In our m-dimensional key distribution, the multicast messages are propagated using directed flooding thorough different quadrants separately. Users are distributed at these quadrants according to the following. For a certain quadrant with *i* 1's, the maximum number of users the quadrant can have is $\binom{k}{k-i}$. Similar to the derivation of (2), the maximum number of unicast messages is $\sum_{i=1}^{\min(k,m)}\binom{m}{i}\binom{k}{k-i}$. With each message containing *k* keys, the message size can be considered as *k* key-size. The upper bound on

the total traffic in the network, T_m, is defined as the maximum number of messages multiplied by the message size. According to the above, this value is determined by (3).

$$T_m = \sum_{i=1}^{\min(k,m)} \binom{m}{i}\binom{k}{k-i} k = k \sum_{i=1}^{\min(k,m)} \binom{m}{i}\binom{k}{i} = k\binom{m+k}{k} \qquad (6)$$

The proof of the T_m final value can be found in Appendix B.

T_m is plotted at different values of k (and accordingly m) in **Fig** 22.

Since every user will decrypt only the keys he or she needs to update, with an extra single decryption overhead per message, the average number of decryptions per user remains as in the simple directed flooding as in (5).

4.1.3. ANALYZING 2D MULTICAST

In the 2D multicast scheme, the group controller generates m messages, one for each of m-unaltered key. Each message contains a concatenation of encryptions for each key of k-altered keys. For LQD nodes in all valid m-length quadrants, the message will be split into k parts, one for each K_j, $1<j<k$. Each part will be sent to all users in the quadrant who

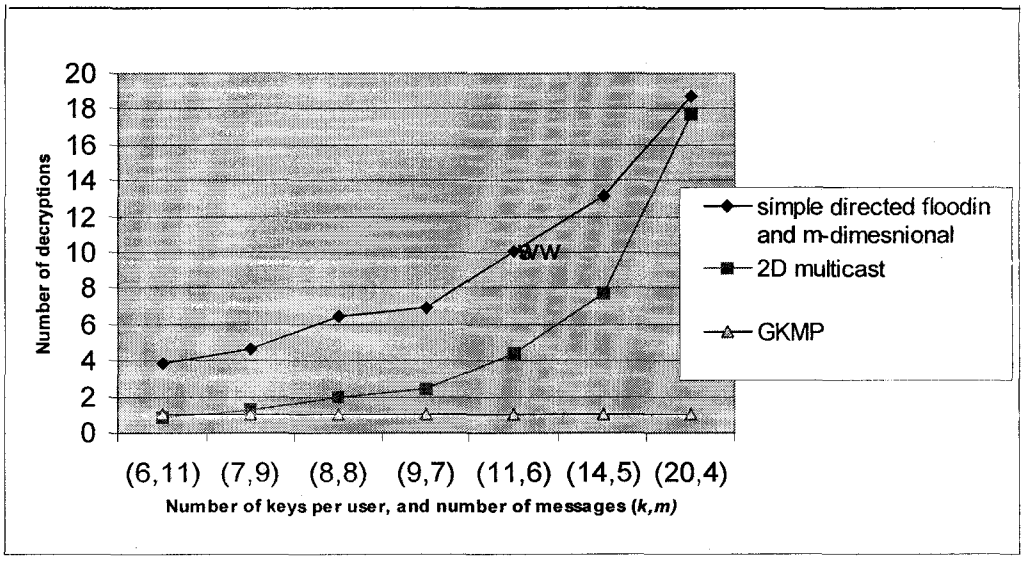

Fig. 21. Average Number of Decryptions per user for 1000 users.

know K_j. In a quadrant who knows i keys out of m-unaltered keys, $0 \leq i \leq m$; there exists a maximum of $\binom{k-1}{k-i-1}$ nodes for each K_j. Since there are $\sum_{i=1}^{\min(k,m)} \binom{m}{i}$ m-length quadrants, the total network traffic, T_{2D} will be determined by (7).

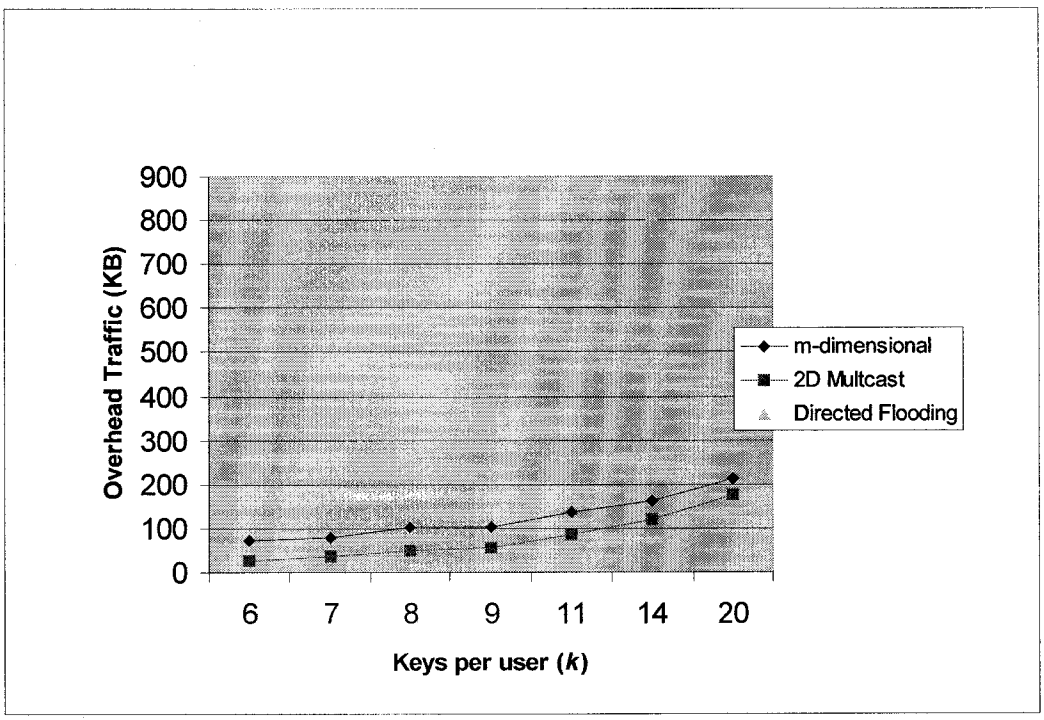

Fig. 22. Total network traffic for 10,000 users under m-dimensional key distribution.

$$T_{2D} =. \sum_{i=1}^{\min(k,m)-1} \binom{m}{i}\binom{k-1}{i} \qquad (7)$$

Fig 22 illustrates the total network traffic for both m-dimensional and 2D multicast as well as the simple directed flooding.

From Fig 20, and **Fig** 22, we notice that using either *m*-dimensional multicast or 2D multicast instead of Directed Flooding significantly reduces the network overhead. The main reason behind this is that in our two schemes users receive only messages needed to update their keys. If we use just Directed Flooding, every user gets a copy of every message.

From Figure 21, it may be observed that the overhead per user is significantly less in 2D multicast than in *m*-dimensional multicast. If we select larger number of keys per user as *k*, the average number of decryptions becomes almost the same in both. This is due to the fact that users decrypt all keys they should know. In case of larger *k*, the number of overlapped keys between any two users is quite large. Thus, each user needs to decrypt a larger number of keys in every rekeying message. In smaller *k*, there is less overlap and users decrypt less number of keys out of every message.

2D multicast uses a one-way hash function to generate the combined decryption key, while m-dimensional multicast uses an extra decryption instead. In the above analysis, we assume a simple hash function that incurs almost no cost with respect to node processing power. For that reason, we neglected such a cost when we selected the number of decryptions per node to represent processing cost. However, this is not usually the situation, especially if we use a more sophisticated and more expensive function.

In **Fig** 23, we notice that the traffic overhead in 2D multicast is almost one third of it in *m*-dimensional multicast. This reduction of overhead traffic is intended by design as every user receives only the portion of keys that she needs. Similar to the average number of encryptions in Figure 21, when the parameter k increases, the traffic overhead comes close to it in *m*-dimensional multicast. This is due to the same fact that when users know more keys, more keys overlap, and accordingly more single key messages per user.

Since both *m*-dimensional multicast and 2D multicast are based on EBS, the key storage requirements for both schemes are the same---*k* keys for individual nodes and *k*+*m* keys for group controller. In **Fig** 24 and **Fig** 25, we plot key storage requirement for GKMP, and all other EBS-based schemes.

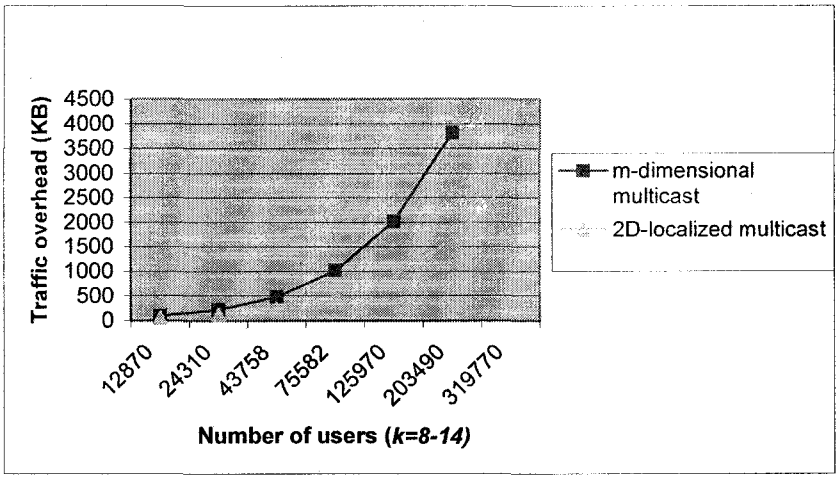

Fig. 23. Scalability of Traffic with changing k.

4.1.4. SCALABILITY ISSUES

The scalability of our proposed approach depends on the scalability of the two underlying schemes, CAN and EBS. Scalability was the main reason for selecting such schemes in the first place. Here, we discuss the scalability of our approach with respect to both storage and communications.

Every node in our space stores simple coordinate information about a maximum of 2*(m+k) neighbors. As number of nodes grows within EBS limit in (1), this factor does not change. Adding nodes more than $\binom{m+k}{k}$ incorporates the choice of adding a new key in either k or m. In such a case, all nodes in the system should accommodate a new dimension to store neighbor information. This incorporates an increase of one record in the storage space incurred on every node. However, the growth of the number of keys is less than logarithmic in terms of N according to [30]. Using an extra key or two for m and k, helps much in avoiding such change in larger networks for the price of a small extra storage. In our case of wireless ad-hoc networks of hundreds to thousands of nodes, this will be enough to maintain storage scalability.

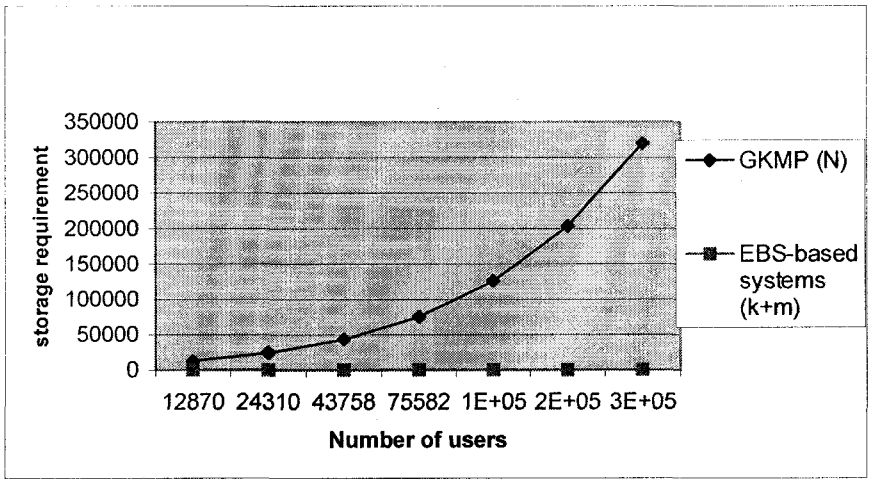

Fig 24. Key Storage Requirement for GC.

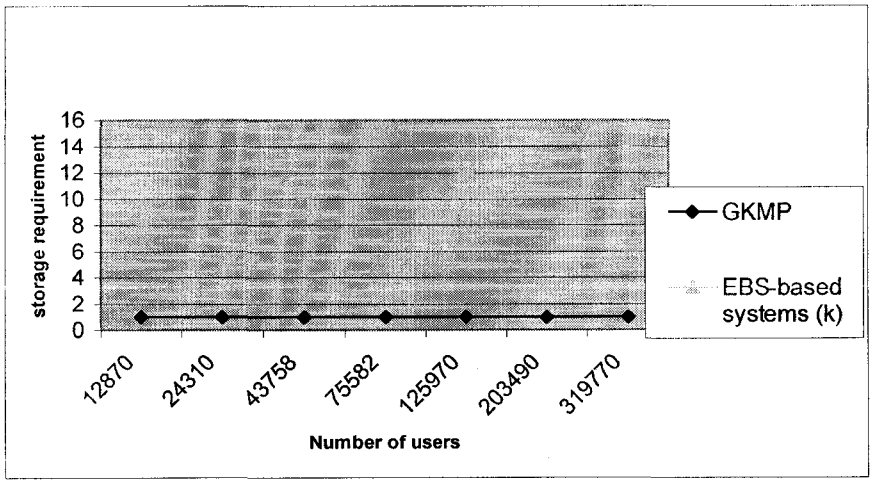

Fig. 25.Ke y Storage Requirement for Individual Nodes.

Our approach assumes a centralized group controller that performs key generation and message construction. Key distribution is done through a completely distributed scheme in both *m*-dimensional multicast and 2D multicast. The communications cost depends on

the parameters m and k. For increasing number of users, adding new nodes within the $\binom{m+k}{k}$ bound results in no change in the communication cost per rekeying. Adding more users than this value results in adding a new key. In such a case, the new key may result in increasing m or k. In both cases the number of messages per rekeying will increase slightly and continues at this new value till we reach the new limit $\binom{m+k+1}{m}$. The new limit is normally more than twice the old limit, so the overhead increases with an order less than logarithmic in terms of N. In **Fig** 23, we plot the traffic overhead for both m-dimensional multicast and 2D multicast when $m=8$ with increasing k.

4.2. ANALYSIS OF TKGS

In this section we analyze the proposed TKGS. We focus on the success probability. To analyze the message cost involved in TKGS, we start by calculating the potential community. According to CKDS [30], each administrative key K_i is known by exactly N_i group members, where

$$N_i=\binom{m+k-1}{m} \tag{8}$$

We assume that TKGS will use directed flooding for all multicast control traffic. For simplicity, we assume a perfectly partitioned space for the community $\{N_i\}$ with respect to K_i. Accordingly, a single broadcast message sent to the whole community will unfold into N_i unicast messages. The total control traffic takes one round of response messages in addition to t-share sending messages. Knowing that most collaborative key management protocols consume multiple rounds to perform rekeying [44], TKGS provides better performance. However, in case of protocol failure, rekeying may consume multiple rounds. In the following we focus on choosing appropriate values for all parameters involved in TKGS to obtain a high probability of first trial success and to achieve the above lower cost.

Given that the individual node failure probability is p, and q=1-p, the probability of success for basic TKGS, S_i, can be computed as follows.

$$S_i = \sum_{j=t}^{N_i} (1-p)^j * p^{N_i - j} \binom{N_i}{j} \quad (9)$$

where t is the threshold at which the dealer proceeds with rekeying of K_i.

Basic TKGP Success probability under t=5% of N_i. In Fig. 26, Fig. 27, and Fig. 28, plotted the probability S_i for the value of the threshold t equal to 5%, 10%, and 20% of N_i, respectively. The figures depict the basic TKGP success probability under different

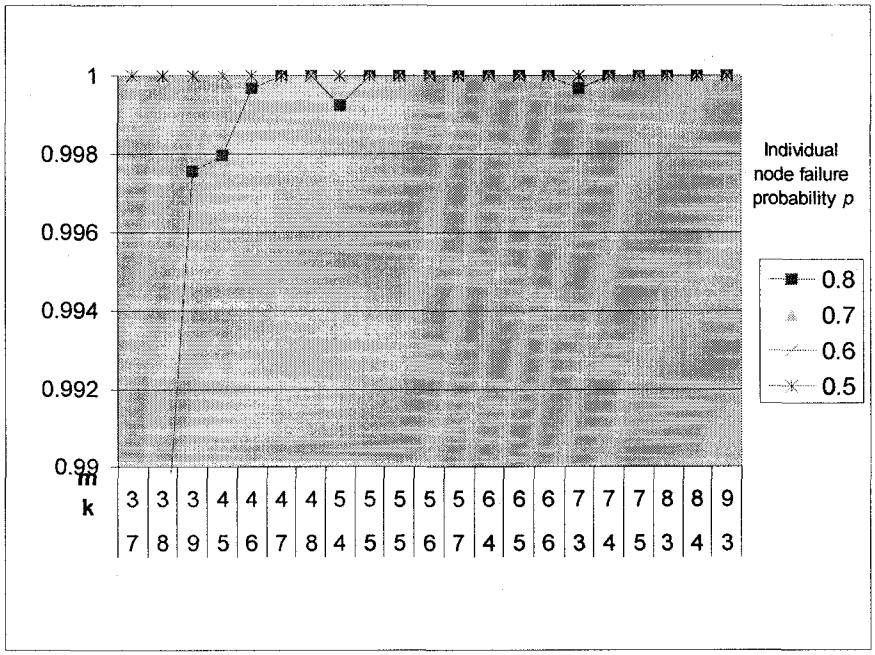

Fig. 26. Basic TKGP Success probability under t=5%of N_i.

values of the individual node failure probability p. Intuitively, the more un-trusted the nodes can be, the less likelihood to find a valid t share. In our environment, we assume the nodes to malicious that the individual node failure probability, i.e. the probability that the node will refuse to participate, send invalid shares, or fail, is as high as 70% or 80%.

Selecting smaller value of the threshold t provide higher likelihood of success. However, the smaller the threshold is, the more vulnerable the system can be under mobile adversary attack. Accordingly, we need to select the threshold t to represent a blocking number of nodes that an adversary needs to control in order to reveal the secret *KGK*.

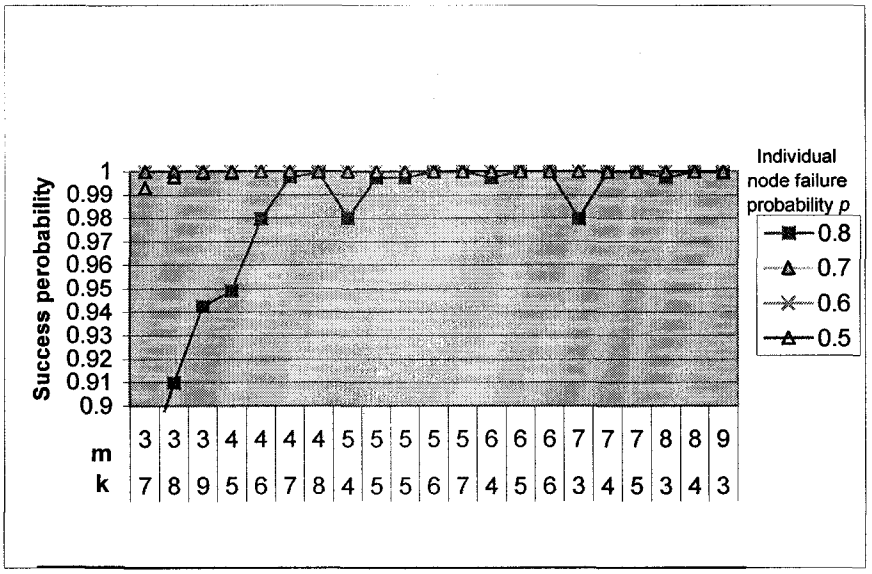

Fig. 27. Basic TKGS Success probability under t=10%of N_i.

Input: EBS(,N,k,m) canonical matrix, c joining users, c

Output: c key string assignment for c users

Step 1: give the first user the string $(11..1)_k(0..0)_m$

Step 2: for zeros=m-1 to 0 do

> *Give each one of C(k+m-zeros,k) user one of the key strings with consecutive zeros zeros at the end.*

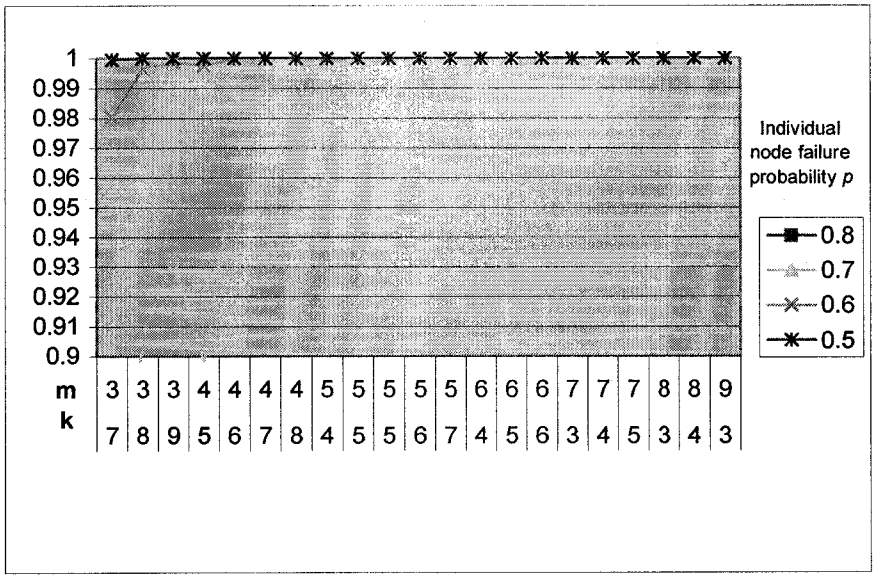

Fig. 28. Basic TKGS Success probability under t=20%of N*i*.

4.3. CKAS ANALYSIS

In this section, we analyze the proposed CKAS. We focus on collusion analysis and computing collusion probability in both EBS and CEBS. Then, we propose a simple key assignment scheme to reduce collusion probability and accordingly, the frequency of rekeying in both EBS and CEBS.

4.3.1. COLLUSION ANALYSIS IN EBS

In EBS, collusion is considered when the set of evicted users know the entire set of keys regardless of whether they communicate to exchange those keys or not. In order for a subset of c users to collude and reveal the system keys, they need to collectively own the whole set of $k+m$ administrative keys. If those users succeed to collaborate before any key update messages are delivered to other users, the system fails. We consider the time between eviction of such users and the update of keys to be a system

vulnerability period. The length of this period depends on the key update rate, the time required to distribute the updated keys and the number of users who need to collaborate, c_{min}. In WAHNs, the first two parameters are greatly affected by the nature of such networks such as high mobility, weak connectivity, hop-by-hop routing, low computational power, etc. The number of users who are capable of such knowledge is greatly determined by the values of m and k. For example, consider the EBS(10,3,2). Any pair of the following users is capable of revealing all the system keys if they collude. (U_1, U_6), (U_1, U_9), (U_1, U_{10}), (U_2, U_5), (U_2, U_8), (U_1, U_{10}), (U_3, U_4), (U_3, U_7), (U_3, U_{10}), (U_4, U_8), (U_1, U_9), (U_4, U_{10}). Simply, it takes the collusion of two users to beat the system. This is due to the fact that the overlap between keys known by different users is quite large. This overlap is maximal when (m/k is close to 1).

In EBS, the parameters m and k represent a tradeoff between number of rekeying messages and number of keys known by each user, if we assume a fixed total number of keys $m+k$. In addition to that role, m and k play an important role in controlling the possibility off collusion through controlling the average number of key overlaps among users. Consider an example EBS(10,3,2), above, the collusion of any pair of users from the set mentioned is enough to reveal all system keys. Since rekeying is not performed individually after each join/leave, leaving users have enough time to collude during the rekeying period before rekeying is performed. Batch rekeying should take place whenever the probability of collusion exceeds a certain threshold. We denote the minimum number of parties needed to collude as c_{min} and the maximum number of parties that might exist without collusion as c_{max}. In the following analysis, we assume an EBS with a complete canonical matrix unless otherwise specified.

4.3.1.1. MINIMUM NUMBER OF COLLUDERS (C_{MIN})

The two key points in reducing the probability of such collusion are selecting the appropriate rekeying period and maximizing the minimum number of users, who need to collude, denoted by c_{min}, in order to beat the system. The two factors are not disjoint. The longer the rekeying period is, the more time is available to leaving users to collude and vice versa. For longer rekeying periods, it is required to have a larger value for c_{min}. However, on one hand, frequent rekeying consumes more communication and

computational overhead. On the other hand designing the system such that c_{min} is quite large implies larger number of rekeying messages. In the following, we analyze the impact of selecting different values of the parameters k and m on the value of c_{min}.

Assuming minimum overlap between user keys gives the maximum contribution to the set of keys revealed by colluding users. For example, in the EBS(10,3,2), Table 1, since each user knows only three keys out of total five keys, the minimum possible overlap is one key. For example, users U_1 and U_{10} have one key overlap, which is K_3. Upon the collusion of U_1 and U_{10}, U_1 contributes K_1, K_2, and K_3, while user U_{10} contributes K_3, which is redundant, K_4 and K_5. The minimum possible overlap is zero keys. A possible example is similar EBS total 6 keys, with 3 keys per user, EBS(20,3,3). In such a system, two colluding users may reveal the system keys by contributing two disjoint sets of three keys each. The maximum possible overlap is $k-1$, when two users have similar keys except one key each. In the following, we explore the theoretical bounds of the required number of colluding parties, c_{min}. The following two theorems set the bounds for the minimum number of colluders, c_{min}, to reveal system keys.

Theorem 1: *In a canonical EBS(N,k,m), if $1 \leq m \leq k$, there exist two users who can collude to reveal all system keys.*

Proof:

Assume there are no two users U_i and U_j who can collude to reveal the $k+m$ keys. Accordingly, if U_i can reveal k keys, and U_j can share $0 \leq k_{ij} \leq k-1$ keys with U_i. If $k_{ij}=0$, then collusion of the two users U_i and U_j reveals $R=2k$ keys, and $2k<k+m$, which means $m>k$, and contradicts the assumption that $m<k$.

If $k_{ij}=k-1$, then $m=1$ and the total number of users is $k+1$. The collusion of the two users U_i and U_j reveals $R=k+1=k+m$, and contradicts the assumption that they cannot reveal the system keys.

If $1<k_{ij}<k-1$, then the collusion of the two users U_i and U_j reveals $R=2k-k_{ij}<k+m$.

Thus, $k-k_{ij}<m$ or $k-m<k_{ij}$, and $k_{ij}>1$, which contradicts with the assumption that $m<k$.

Theorem 2: *In a canonical EBS(N,k,m) with N=C(k+m,k), the minimum number of users required to collude, c_{min}, is at most m+1.*

Proof:

The first colluding user contributes *k* keys to the set of revealed keys, R. Since each user has a distinct key assignment, the maximum contribution to the set R is 1 key. Accordingly, to reveal *m* remaining keys after the first colluding user, we need at most *m* users. Thus, the number of colluding users is at most *m+1*.

From the above discussion,

$$m+1 \geq c_{min} \geq 2 \qquad (10)$$

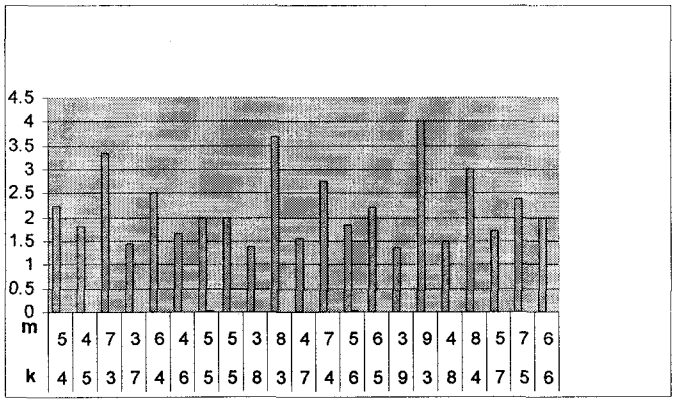

Fig. 29. Min number of colluding parties required to reveal system keys, c_{min}.

In order to select the values of *m* and *k* to set a specific value for the parameter c_{min}, we assume the worst case of no overlap. In such case, each colluding user will reveal *k* keys, and the collusion of c_{min} should not reveal the total *m+k* keys, i.e., k+m≤kc. If we add the result from (10), we get

$$1 + \mathrm{m/k} \prec c_{\min} \leq N \qquad (11)$$

Thus, selecting *m* and *k* such that (11) is satisfied will guarantee a required minimum number of parties to collude in order to reveal the system keys.

Applying the above to our EBS(10,3,2), we have *m/k*=2/3. Since 0<2/3≤1, our minimum number of colluding parties, c_{min}, is 2.

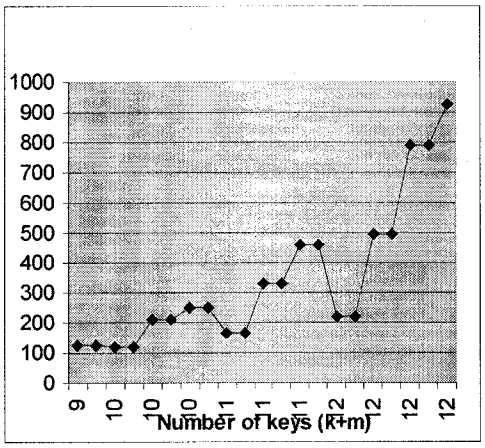

Fig. 30. Max Number of supported users for different selections of m and k.

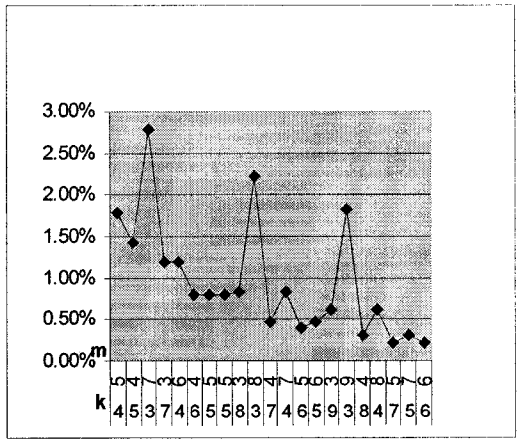

Fig. 31. Percentage users required to collude , e.

In the above example, we selected m and k to produce c_{min} equal to either two or three parties out of total number of users N=10. Although these values represent 20-30% of total number of users, they tend to be unpractical for larger number of users. In a population of N users, with frequent rekeying, selecting m and k to produce values of c_{min} that reflects 3 to 4 % of total population is reasonable. **Fi** 29 depicts possible selections of values for m and k to support a population of N=100, and different values of c_{min}.

We assume the number of colluding user to be a percentage, e, of the total population, N. Accordingly, we have

$$C_{\min} = \left\lceil 1 + \frac{m}{k} \right\rceil \prec eN \tag{12}$$

However, in a canonical EBS, the maximum number of users, N_{max} is determined as follows.

$$N \leq N_{\max} = \binom{k+m}{k} \tag{13}$$

As in Figure 30, we select k and m such that they produce an acceptable value of c_{min}, even if they produce a value of N_{max} larger than our projected group size N. If we consider N_{max} instead of N in (11), and considering N_{max} quite large, we get

$$e_{\min} \succ \frac{1}{\binom{m+k-1}{m}} \tag{14}$$

This percentage is depicted in Fig 31. Due to the different number of users supported by different selections of m and k, the percentage off colluding users e, does not correspond directly to the value of c_{min} in Fig 31.

4.3.1.2. MAXIMUM NUMBER OF NON-COLLUDERS

Having a number of c evicted users in EBS implies that there is a chance of collusion. Intuitively, the larger the number of potential colluders, c, the more likely the collusion can happen. If the number of colluders is 1, clearly, she can know no more than k keys and cannot collude as long as there are some other keys unknown to her (m>0). On the other hand, if we have all users to collude (c=N), it is certain that those colluders know all the keys. The following theorem determines the maximum number of potential colluders who are unable to collude, c_{max}.

Theorem 3: *In an EBS(N,k,m), if 1≤m≤k, there exist up to* $c_{\max} = \binom{k+m-1}{k}$ *users who collectively do not know all the keys.*

Proof:

For a group of c_{max} users to collectively do not know the entire set of k+m keys, there exist at least one key, K_u, unknown to all the members of the group. Accordingly, all c_{max} users have a zero in their key string in the position u. Their key strings represent all possible combinations to distribute k keys in the remaining k+m-1 positions. Thus, $c_{\max} = \binom{k+m-1}{k}$.

A result of theorem 3, if we have c_{max}+1 potential colluders or more, collusion is certain. The value of c_{max} sets an upper limit on the number of evicted users (and potential colluders) that can be allowed before rekeying. In other words, if at any time there exists c_{max}= evicted users, rekeying is a must. Otherwise, any other eviction will certainly cause collusion.

As an example of the above theorem, consider the key assignment of EBS(10,2,3) in table 2. We can collect up to 6 users, $\binom{2+3-1}{2}$, such none of them know a specific key, e.g. K_5. Those users are U_1, U_2, U_3, U_4, U_7, and U_9. Adding any other user to this collection will reveal k_5 and cause collusion.

Probability of collusion

According to the above, having c_{max} an evicted user is pretty dangerous. Rekeying should take place quite long before the number of evicted users reaches this value. The question is how to quantify the possibility of collusion when having a number of c potential colluders so that rekeying takes place if this probability is larger than a certain threshold, say 0.5. In the following we suggest an approximate function that quantifies this probability.

For a set of c colluders we compute the probability that they know a set of k+m keys, p. In case $c \leq c_{min}$, this probability is zero since at least c_{min} users are needed to reveal all k+m keys. In case $c > c_{max}$, the probability evaluates to one (a certain collusion) as described above.

In all cases $c_{min} < c \leq c_{max}$,if we assume the number of colluders , c, is significantly less than the total population N, the collusion probability may be approximated as follows. The probability of not knowing these keys (1-p) is the probability that there exist one key (or more) unknown to all users.

The probability that the first user does not know a certain key, k_u , $p(1,k_u)$ is

$$p(1,k_u) = \frac{\binom{k+m-1}{k_c}}{\binom{k+m}{k_c}}. \qquad (15)$$

The probability that the second user does not know the same key , k_u , given that the first user does not know the key p(2,ku)is

$$p(2,k_u)=\frac{\binom{k+m-1}{k_c}-1}{\binom{k+m}{k_c}-1}. \tag{16}$$

Generally the probability that the i^{th} user, $c \geq i \geq 1$, does not know k_u, p(i,ku),given that the all previous users do not know the key is:

$$p(i,k_u)=\frac{\binom{k+m-1}{kc}-i+1}{\binom{k+m}{kc}-i+1} \tag{17}$$

The probability that all the c users do not know the specific key, k_u, can be independently computed as follows

$$p_c=\prod_{i=1}^{c}\frac{\binom{k+m-1}{kc}-i+1}{\binom{k+m}{kc}-i+1} \tag{18}$$

And accordingly, the probability of the set of c users collectively knows the key k_u is 1-pc. The probability that they know all the k+m keys, p, can be computed as

$$p=\left(1-\prod_{i=1}^{c}\frac{\binom{k+m-1}{kc}-i+1}{\binom{k+m}{kc}-i+1}\right)^{k+m} \tag{19}$$

To verify the accuracy of this approximate probability function, we consider the cases of number of colluders, $c<c_{min}$. In such case, we expect the probability of collusion to be zero. The function evaluates to a very tiny value close to zero. In case number of colluders more than c_{max}, the function evaluates to one.

Selecting the values for k and m as the design parameters of an EBS system affects the values of both c_{min} and c_{max}. The probability of collusion of randomly selected c users is also affected with the selection of these two parameters and accordingly the frequency of rekeying. Figure 32 depicts the effect of selecting different values of k and m on the probability of collusion for a 100 users.

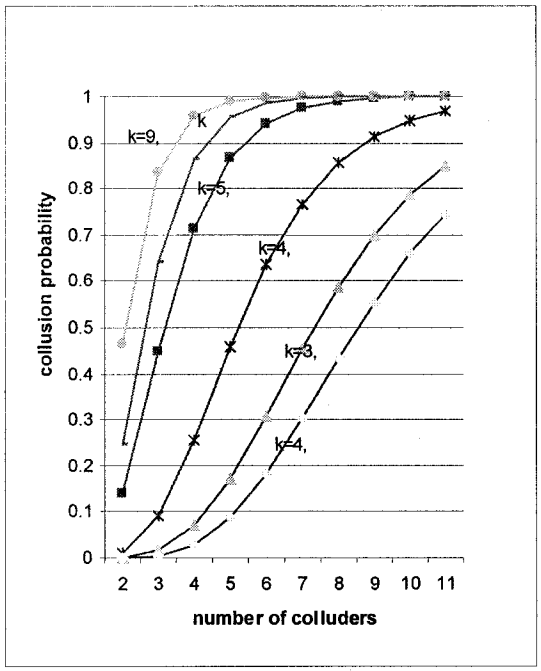

Fig. 32. Collusion probability for different EBS key assignments for 100 Nodes.

From the figure, it can be seen that EBS systems with larger values of the ratio *k/m* implies a lower collusion probability of a specific number of colluders, *c*.

4.3.2. COLLUSION ANALYSIS IN CEBS

In clustered EBS, for collusion to take place, the colluding parties must cover both k_c+m_c cluster keys as well as all k_g+m_g global keys. The colluding parties have to span at least $c_{min,g}$ clusters to cover the set of k_g+m_g. Assuming a controllable inter-cluster communications, this is very unlikely to happen. In case inter-cluster communications is there, the collusion probability can be computed the same way we derived (17). In such case, we are considering collusion in a 2 layer independently cascaded EBSs. For independence, the collusion probability in a clustered EBS, $p_{c,}$ can be expressed as the product of two independent EBS collusion probabilities. Under the same assumption for EBS, the probability might be approximated as follows.

$$p = \left(1 - \prod_{i=1}^{c} \frac{\binom{k_g+m_g-1}{k_g} - i + 1}{\binom{k_g+m_g}{k_g} - i + 1}\right)^{k_g+m_g} \left(1 - \prod_{i=1}^{c} \frac{\binom{k_c+m_c-1}{k_g} - i + 1}{\binom{k_c+m_c}{k_c} - i + 1}\right)^{k_c+m_c} \quad (20)$$

In CEBS, for the collusion to take place, at least $c_{min,c}$ parties from at least different $c_{min,g}$ clusters should exchange keys. In case of location-based clustering with no inter-cluster communication, the collusion probability is next to zero. On the other hand, if we have logical clustering with allowed inter-cluster communications, the collusion probability can be determined by (20). Fig. 33 show that having 1000 nodes logically clustered even on random basis achieves lower collusion probability than using plain EBS.

4.4. CONCLUSION

In this chapter, we compared our proposed architecture for key management with existing solutions. In CKDS, we combined two scalable schemes to introduce an efficient application-level key management protocol for secure group communication in wireless ad-hoc networks. We used CAN for application-level multicast and EBS for key distribution. We assumed a centralized group controller that performs key generation and rekeying message construction. We introduced two new key distribution schemes, *m*-dimensional multicast and 2D multicast to deliver keys efficiently to group members in such an environment.

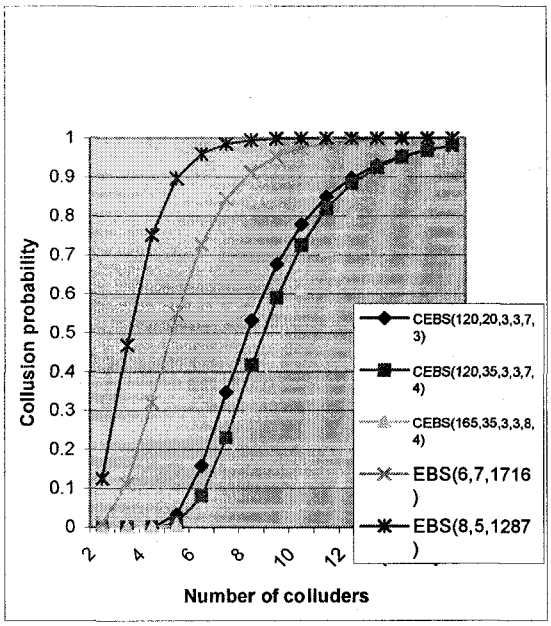

Fig. 33.Col lusion probability in supporting a 1000 users with EBS and CEBS.

We used Directed Flooding as a multicast message forwarding technique in our approach at individual quadrant level. We compared our scheme to using this type of flooding solely to deliver keys.

For key generation we evaluated the proposed TKGS. TKGS was founds to achieve a significant network traffic reduction of almost one eighth of it in using normal application-level multicast in CAN. We compared storage requirement and overhead incurred on each node to GKMP. Our scheme showed better scalability and lower storage requirement at the cost of small number of decryptions performed by each individual node. We showed that selecting lower number of keys per user, and accordingly higher number of rekeying messages, in EBS achieves better performance under our 2D multicast key distribution scheme. We showed our approach to be scalable with respect to

both storage and communication. We believe this approach is suitable for mid size wireless ad-hoc networks with hundreds or thousands of users.

For key assignment, we proposed CKAS and CEBS that minimizes the collusion probability of co-located nodes. We evaluated our proposed techniques and presented a comprehensive group management module, an efficient decentralized scheme that performs key assignment, generation, and distribution in compliance with this architecture. We overviewed each of these components and showed how they meet the architectural objectives. We performed some analysis on the performance parameters of each component separately and showed the feasibility of such design compared with existing architecture. Currently, we are integrating all the key management and group session management components together and define all the interaction. Our next step is to perform simulation experiments on such integrated group management scheme and compare real network parameters, such as actual delays and traffic overhead, with other existing key management schemes.

CHAPTER V

KEY MANAGEMNT IN WIRELESS SENSOR NETWORKS

The envisioned growth in using sensor networks in a wide variety of applications ranging from healthcare to warfare is fueling extensive research in securing these networks. We have an unprecedented window of opportunity to build this emerging technology right! Key management is crucial to the secure operation of sensor networks. A large number of keys need to be managed in order to encrypt and authenticate all sensitive data exchanged. The characteristics of sensors and sensor networks, including lack of physical protection and the resource constrained nature of sensors render most existing key management solutions developed for other networks (for example, PKI-based solutions) infeasible for sensor networks. The tradeoff between managing acceptable levels of security and conserving network energy for sensor network operation is a challenging task.

Recently a number of key management schemes have been developed for sensor networks [25] [14] [122] [32] [121] [56] [17] [27] [134] [132] [29] [33]. We broadly classify key management schemes for sensor networks into static and dynamic keying based on whether the administrative keys (those used to establish communication keys) are updated or not after the initial network deployment and setup. A well known static keying scheme is due to *Eschenauer* and *Gligor* [35]. In this scheme, each sensor node is assigned k keys out of a large pool P of keys in the pre-deployment phase. Neighboring nodes may establish a secure link only if they share al least one key, which is provided with a certain probability based on the selection of k and P. A major advantage of this scheme is the exclusion of the base station in key management. However, successive node captures enable the attacker to reveal network keys and use them to attack other nodes. An enhancement to this scheme was proposed in [17] in which two nodes can establish a link only if they share q keys. Liu and Ning [70] provided further enhancement by using t-degree bi-variate key polynomials. Since an attacker needs to capture at least $t+1$ nodes to obtain any *t-degree* polynomial, this solution was shown to significantly enhance network resilience to node capture as long as the number of

captured nodes is below a certain threshold (around 3% as shown in[70])\. However, if the number of captured nodes exceeds this threshold, the network is almost entirely captured by the attacker.

Another problem with the abovementioned static key pre-distribution schemes is the reduction in network connectivity resulting from the use of a large key pool, *P.* As P increases for the same k, network connectivity decreases since only nodes that overlap in one or more keys (or key polynomials) can directly interact. Unlike static keying, dynamic keying schemes change all keys revealed to an attacker upon node capture. The major advantage of dynamic keying is enhanced network survivability, since any captured keys are replaced in a timely manner. Also, when adding new nodes, unlike static keying, the probability of network capture does not necessarily increase.

Jolly et al. [56] have proposed a dynamic key management scheme that is using Identity Based Symmetric Keying. Although their approach requires very few keys to be stored at each node, the re-keying procedure is inefficient due to the large number of messages exchanged for key renewals. In addition, they require a centralized key server to play a major role in key management. Another group of dynamic keying schemes, including the one proposed in this thesis, are based upon Exclusion Basis Systems (EBS) - a combinatorial formulation of group key management problem [30]. In EBS-based schemes, each node is assigned k keys out of pool of size $k+m$ keys. Once one or more nodes are captured (or suspected to be captured), rekeying takes place by generating replacement keys, encrypting them with all the m keys unknown to the captured nodes and distributing them to the other nodes. However, since the value of m is selected to be relatively small, to make rekeying feasible in terms of number of messages, a small number of nodes may collude and collectively reveal all the network keys. EBS-based schemes with collusion-resistance have been proposed recently in [14] [31] [121]. While more energy efficient than Jolly et al's scheme, both schemes still considerably rely on a centralized key server to perform rekeying.

The main contribution of this chapter is proposing an efficient and scalable key management scheme for clustered long-lived sensor networks. In order to prevent the attacker from knowing the keys by successive node capturing, we propose the use of key polynomials. The rest of this chapter is organized as follows. Section 1 describes our

basic Dynamic Combinatorial Keying (DCK), the dynamic keying scheme for sensor networks. Section 2 describes the use of key polynomials to replace the keys. Section 3 describes our proposed solution. Section 4 describes the attack mitigation. Finally, Section 5 describes the use of key polynomials with our proposed solution.

5.1. STATIC VERSUS DYNAMIC KEY MANAGEMENT IN SENSOR NETWORKS

As described in the introduction, static key management schemes depend on the pre-distribution of a randomly selected set of k keys [35] (or bi-variate key polynomials [71]) to each node out of a pool of $P=k+m$ keys. Two nodes can communicate directly if they are within the transmission range of each other and they share at least one key/key polynomial. Since the key polynomial model is more general, it is used in the analysis below. For simplicity, we only consider key management within a cluster.

The probability of sharing a key polynomial between any two randomly selected nodes P_s, is defined as follows:

$$P_s = 1 \qquad \text{if } k>m,$$

$$P_s = 1 - \prod_{i=0}^{k-1} \frac{m-i}{k+m-i} \qquad \text{if } k \leq m, \qquad (21)$$

where k is the number of polynomials known to a node and m is the number of polynomials unknown to that node.

It two nodes are neighbors, they can directly communicate only if they can establish a communication key, otherwise, they use an intermediary node that shares at least one key polynomial with each of them to establish an indirect communication key. The network topology as well as the transmission range determines the physical network connectivity, whereas the probability P_s is an indicator for logical connectivity.

Static keying schemes select a large value of k (e.g. 250 or more [35]) to guarantee reasonable connectivity (e.g., $P_s > 0.99$). However, since the value of k is directly related to the node storage capacity C, it might not be a design option to use such a large value of k, especially in case of polynomials where for polynomial degree t, $C=(t+1)k$. On the other hand, in order to enhance the network resilience to node capture, m is selected to be

quite large so that more nodes would be needed to reveal all keys. However, the larger value of m leads to less network connectivity since the probability of sharing a key is lowered. The relationship between resilience, connectivity, and m in a 20,000 network is shown in the next section. Since LOCK (and other EBS-based schemes) essentially use a similar idea for selecting k polynomials for each node as in [71], formula (21) holds for these schemes too. However, dynamic schemes operate at lower values of m, thus guaranteeing higher connectivity in general.

The probability P_{Fi} of compromising a specific key polynomial, F^i, given the capture of N_c nodes that communicate directly is given by (22)

$$p_{F_i} = 1 - \sum_{i=0}^{t} \binom{N_c}{i} \left(\frac{k}{k+m}\right)^i \left(\frac{m}{k+m}\right)^{N_c - i} \qquad \text{for } N_c > t$$

$$P_{Fi} = 0 \qquad \text{otherwise} \qquad (22)$$

The probability P_c of any direct or indirect key to be compromised is

$$P_c = P_s * P_{Fi} + (1-Ps)\,(1-P_{Fi})^2)\,((1- N_c/N)\,, \qquad (23)$$

In [71] these probabilities were studied against the polynomial degree t. It was shown that network resilience is maintained up to a certain threshold N_c^* of captured nodes, with probability very close to zero. When N_c exceeds N_c^*, the probability rises rapidly very close to one. When P_c is close to one, then capturing a small number of nodes may reveal all keys, i.e., the network is fully captured and controlled by the attacker. Fig. 34 shows the network capture probability against N_c. The network capture point N_c^* is defined as the number of captured nodes at which the capture probability is close to one. N_c^* is considered a measure for network resilience.

Different parameters affect the value of N_c^* including k, m, and t. First, increasing the polynomial degree t enhances the network resilience at the expense of storage per node. Simple keys can be considered as polynomials of degree zero. Using simple keys, capturing a very few number of nodes is usually enough to capture the network as long as the attacker is assumed to foster collusion among these nodes without being restricted to the node locations. Secondly, assigning more polynomials to each node (i.e. using a larger value of k) enhances the connectivity (i.e. the probability of sharing a polynomial). However, since more polynomials are known by each node, this also increases the number of polynomials revealed to an attacker once a node is captured, which certainly

decreases the network resilience (N_c^*) as shown in Fig. 34. Finally, increasing the size of the polynomial pool while using the same number of polynomials per node (i.e. increasing *m)* tends to enhances resilience since more polynomials will be needed to capture the network. However, this solution reduces connectivity since the probability of sharing a polynomial falls.

LOCK attempts to perform re-keying early enough before the number of captured nodes reaches N_c^*. Since the node capture probability rises rapidly close to N_c^*, re-keying can be performed at the rising edge of the node capture probability; the point termed R-point in Fig. 49. The re-keying overhead for doing that is sending m messages, each containing k polynomials. One obvious way to minimize the number of times re-keying is needed is to enhance the network resilience (increasing N_c^*) by using a lower value of m and a quite larger value of k. Since re-keying is performed early enough before reaching N_c^*, we can afford to use a smaller value of t to save storage without jeopardizing network security.

In conclusion, from the above analysis, both static and (EBS-based) dynamic schemes share the idea of selecting a random subset of keys for each node out of a pool of keys. Static schemes tend to rely on using a larger key pool to enhance network resilience to attacks, whereas dynamic schemes use a limited pool of keys as well as a limited number of keys per node to achieve better network connectivity. In the latter schemes resilience to attacks is primarily achieved by re-keying. Table 2 summarizes primary features of static and dynamic key management schemes and provides a qualitative comparison of both classes of schemes.

TABLE 2.
Summary Comparison of Static and Dynamic Key Management

	Static keying	Dynamic keying
Network life	Assumed short-lived.	Assumed long-lived.
Key pool	Very large pool; Static administrative key values.	Small size pool; dynamic administrative key values.
Key assignment	Once at pre-deployment	Multiple times post deployment
Key generation	Once at pre-deployment	Multiple times post deployment
Key distribution	All keys are pre-distributed to nodes prior to deployment	Subsets of keys are re-distributed to some nodes as needed
Handling node capture	Revealed keys are lost	Revealed keys are altered ($k \rightarrow k'$)
Re-keying cost	May be practically infeasible with respect to number of messages (m is very large, e.g., 20,000).	Requires few messages (m is small, e.g., 20).
Communication cost	Not applicable for administrative keys (key pre-distribution).	Re-keying overhead.
Storage cost	More keys per node.	Fewer keys per node.
Handling node addition	New node preloaded with keys from static pool – may decrease network resilience to node capture.	New node receives new set of keys; other nodes may be re-keyed – less impact on network resilience to node capture.
Network resilience	High as long as number of nodes captured is small (assuming key polynomials). Once a threshold is exceeded, resilience falls sharply.	High, largely independent of number of nodes captured as long as re-keying is performed timely.
Network connectivity	Less connected due to large key pool. Connectivity improves with increasing number of keys per node.	More connected due to small-size key pool.

5.2. OVERVIEW AND FUNDAMENTAL DESIGN PRINCIPLES OF DCK

Our goal is to efficiently mitigate node capture attacks in long-lived large-scale sensor networks operating in hostile environment. The network should maintain freshness of keying material and be able to survive multiple node captures. Our design decisions are guided by this goal.

Nodes are preloaded with certain code (*initial state)* prior to deployment. All nodes belonging to the same class should carry the same code initially with no individual node-specific pre-programming. No distinctive state per node is needed (compare with static key-pre-distribution schemes). Communication among nodes is encrypted with regularly-updated communication (or session) keys. Administrative keys are used to generate and update session keys. DCK manages both administrative and session keys. Node capture attacks, including the capture of sensor nodes and cluster leaders, are handled through different levels of rekeying (or changing administrative keys). Because the sensor network is assumed to be long-lived and operates in hostile environment, DCK should survive continual attacks.

DCK is based on EBS for efficiency. EBS was shown to out-perform other dynamic key management schemes (such as LKH) in terms of storage and communication overheads [16]. In addition, EBS-based dynamic keying schemes were shown to outperform static keying schemes (e.g. pair-wise key pre-distribution [20]) in terms of network [23]. In order to achieve scalability with respect to both the number of sensor nodes and the number of clusters, DCK uses two cascaded EBS systems at two different levels to manage both administrative and session keys.

As we proposed in [85], key management can be split into three basic sub-tasks, namely, key assignment, key generation, and key distribution. In DCK, we separate the concerns of such duties from the physical architecture of the network. We use three types of nodes, namely, Key Assignment Node (KAS), Key Generation Node (KGN), and Key Distribution Node (KDN). This separation enhances security, robustness and flexibility of the system. Dynamically assigning key management functions to different entities in the system helps in mitigating attacks as well as adaptation to meet certain constraints.

The normal network operation involves data processing and forwarding on the one hand and key management on the other hand. DCK deliberately assumes that data processing may be independent of key management. As stated earlier, cluster leaders are not necessarily data aggregation and forwarding nodes. Separation of concerns may enhance the security since the attacker will not be able to target both processes by capturing specific nodes. For example, a captured cluster leader does not necessarily provide the attacker with the current data aggregation results in the cluster. Another advantage is allowing the altering of data processing topology without disturbing the key management process during the network operation and vice versa.

We start with an overview of EBS, the theoretical basis of our solution, followed by an overview of DCK's network components.

5.2.1. DCK COMPONENTS

DCK uses two layers of EBS administrative keys. The upper layer (level 1) is EBS_b that enables the base station (BS) to manage the cluster leaders as a group. The main motivation for using EBS in the upper layer is for its efficiency and scalability in terms of number of clusters. Recall that we assume that the capture of a Cluster Leader (CL) is as likely as any other sensor node. The EBS_b administrative keys are used to construct group session keys used by the base station to communicate with cluster leaders.

The lower layer (level 0) involves an EBS_{Ci} for each cluster C_i. The clus ter leader is a member in both the upper EBS_b as well as the lower EBS_{Ci}. EBS_{Ci} administrative keys are also used to construct cluster session keys used by the cluster leader to communicate with the sensor nodes within the cluster. The CL is considered a regular member in EBS_C that knows as many cluster administrative keys as any other node in the same cluster (k keys out of k+m keys). Accordingly, the capture of a CL does not provide the attacker with any more cluster keys than the capture of a regular sensor node. Fig. 34 and Fig. 35 shows the physical and functional components of DCK respectively.

During the initialization phase, the sensor nodes in each cluster establish a set of backup keys (one chain of keys for each cluster) shared with the base station and unknown to their (or any other) cluster leader. Key generation of EBS_c keys is performed by a group

of sensor nodes within the cluster (including the cluster leader) called Key Generation Nodes (KGNs). Key distribution is performed by the cluster leader.

The capture of a sensor node (or a KGN) is handled solely by a local re-keying mechanism within the cluster to exclude the captured node from EBS_C. Handling the capture of a cluster leader is done through re-keying of both EBSs in order to exclude the captured cluster leader. Deploying a new cluster leader for a cluster whose leader has been captured requires the cluster sensor nodes to authenticate the new leader using the backup keys shared with the base station.

DCK does not involve any inter-cluster leader communication to generate new cluster keys as in SHELL [122] and [14]. We will show through security and performance analysis that DCK consumes less energy than the other EBS-based schemes without jeopardizing the network security. In the remainder of this section and in the next two sections we describe physical and functional components of DCK respectively.

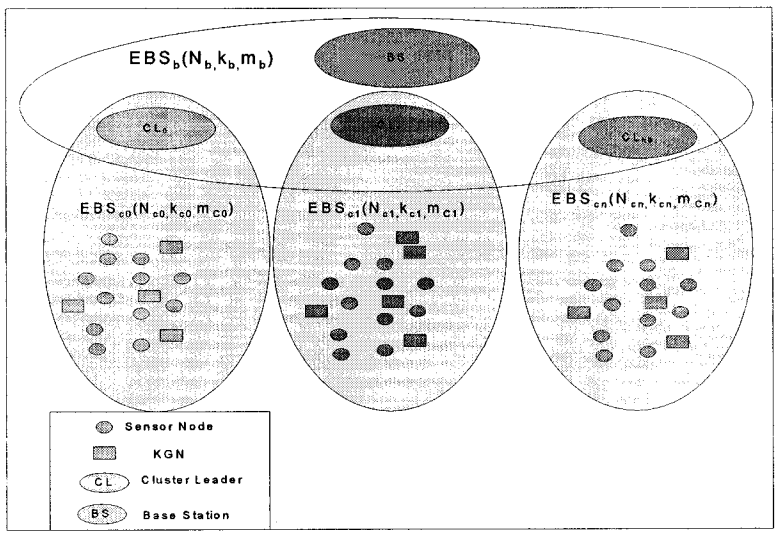

Fig. 34. Physical components of DCK.

5.2.2. DCK NETWORK COMPONENTS

The components of the DCK network components are: the base station (BS), cluster leaders (*CLs*), and the sensor nodes (*SNs*). Nodes are organized in clusters as well as communication groups. Each communication group uses an exclusion basis system to manage their administrative and communication keys. Fig. 34 depicts this logical and

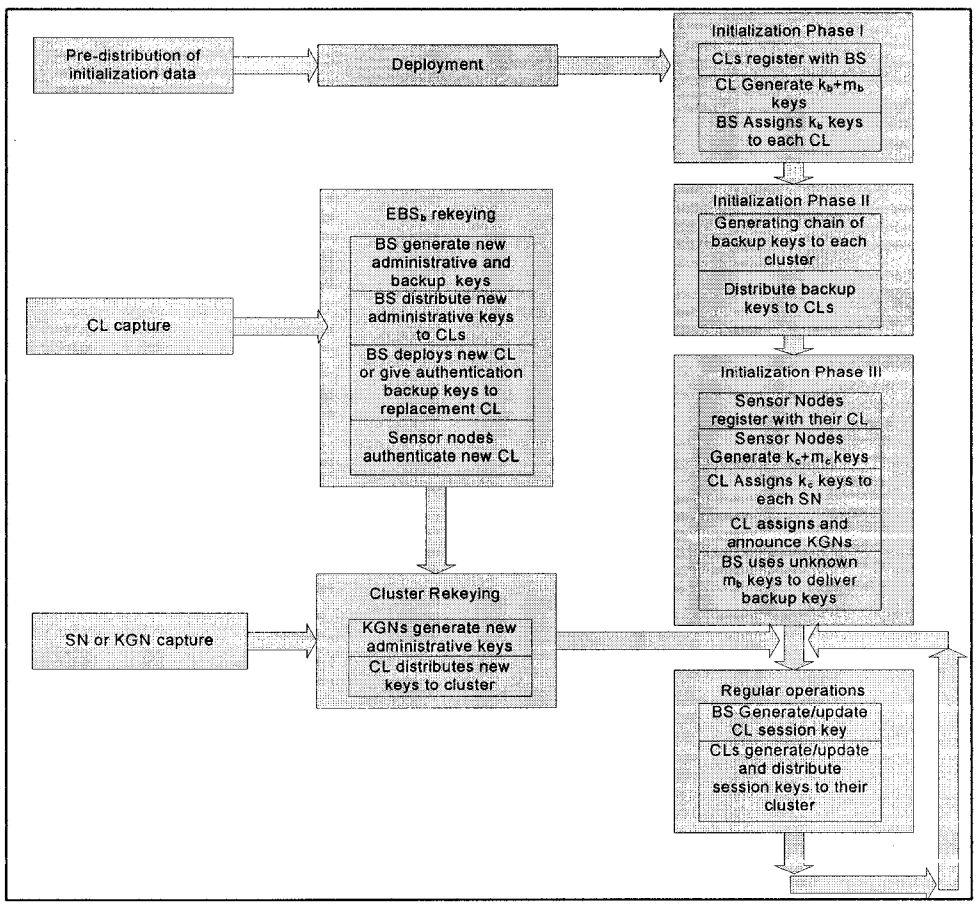

Fig. 35. DCK Functional Components.

physical organization of network components. In the following, we briefly state some assumptions and characteristics of these components.

Sensor Nodes (SNs): are very limited in communication, computation, and storage capabilities. Each *SN* has a pre-deployment unique *id*, a pre-deployment *initial state*, which includes two one-way key generation functions KGF_c and KGF_b, as well as a deployment seed S_b. Each *SN* is equipped with a short range radio as well as an optional location sensing device, such as a GPS receiver (only if required by the collusion mitigation algorithm [27]. *SNs* are assumed to be stationary once deployed, and have short duty cycles to save energy. They are capable of transmitting and receiving short radio packets within their communication range. No individual shared key is assumed between a SN and the base station or cluster leader nodes. During the bootstrapping, the base station establishes backup keys with the sensor nodes of each cluster. Thos backup keys are unknown to the cluster leaders. A cluster leader node, CL_{ci} manages each cluster, c_i, through a cluster-specific exclusion basis system $EBS_{ci}(N_{ci},k_{ci},m_{ci})$. A number of *SNs* within a cluster, specifically k_{ci} nodes at each cluster, are appointed as Key Generation Nodes (*KGNs*).

Cluster Leaders (CLs): A *CL* is a node designated to manage other nodes in the same cluster. CLs may have better capabilities than sensor nodes including better, but limited, batteries, storage and computation capabilities. *CLs* may have a location sensing facility. Cluster leaders may be dynamically assigned to their clusters, which may lead to network re-configuration to mitigate attacks. In DCK, cluster leaders are not necessarily data gateways or data aggregation nodes as in other schemes [14] [122]. CLs are considered key gateways and may have nothing to do with data aggregation. One important reason for this distinction is the generic nature of DCK. The separation between key management and other sensor network functionality enables DCK to serve at different network configurations independent of the data-related network organization.

Currently, we assume *CLs* to be stationary. However, we are studying the mobility of *CLs* to meet application and network security requirements. For scalability reasons, *CLs* do not share individual keys with the base station. They are managed by the base station through a $EBS_b(N_b,k_b,m_b)$. Each CL has a pre-deployment unique id, a pre-deployment *initial state*, which includes a one-way key generation function KGF_b. A CL does not

store (or know) the sensor node key generation function or the k_c+m_c EBS keys used to manage its own cluster. However, CLs are capable of storing the EBS matrix for their clusters with key ids as a row header and node ids as a column header similar to 1. The same column header is also used as small registry to record the IDs and locations of all nodes in the cluster. The CL duties include:

- Registering all sensor nodes in the cluster
- Playing the role of a Key Assignment Node (*KAS*) by generating the EBS matrix and assigning key strings to individual nodes.
- Appointing specific sensor nodes to play the role of Key Generation Nodes (*KGS*)
- Performing the role of a Key Distribution Node (*KDS*) by broadcasting rekeying material to all cluster nodes. In case such facility is not available in the *CL*, another node in the cluster that has such facility might take that role. Alternatively, a flooding protocol might be used.

Base station (BS): The BS is the chief executive entity of the sensor network. It delegates all data sensing, aggregation and fusion to it subordinate *SNs* and *CLs*. In this work, we focus on the key management role of the BS. The BS is assumed to be a trusted entity and cannot be compromised. The following are the key responsibilities of the BS:

- Acts as a central repository for valid node IDs and *KGF* seeds.
- Admits the *CLs* and *SNs*, initially deployed as well as latently added to the network after.
- Manages the *CLs* through EBS.
- Deploying and setting up new CLs and/or redistributing SNs to clusters.

It is worth mentioning that the BS does not intervene in regular network key management functionality, including periodic rekeying within the clusters and handling the capture of a SN. In the capture of a CL, the BS intervenes to maintain the security of network keys.

5.3. DESCRIPTION OF DCK

As shown in Fig. 35 DCK functional components include pre-deployment and post-deployment initialization, regular network operations as well as attack mitigation procedures. Starting with pre-deployment phase, we devote this section to discuss each of these functional components and how they meet the design objectives.

5.3.1. DCK INITIALIZATION

5.3.1.1. PRE-DEPLOYMENT INITIALIZATION

The pre–deployment phase securely implants the initial state in all nodes. One major advantage of DCK is that all SNs are preloaded with the same state except for a unique ID. A similar approach was followed in [29] with nodes carrying no unique IDs. A valid node ID is a long enough bit string with some parity code to rule out random numbers. In addition to the unique ID, each node is also preloaded with two key generation functions, KGF_b and KGF_c.

A key generation function is a secure one-way hash function that takes two parameters, typically a key and a data string, and produces a unique string based on the input. In the context of this thesis, we call such two parameters: a seed (S); and a key (K). Popular message digest functions such as SHA1 [24]or MD5 [25] may be used for such purpose. In order to generate a key, each node applies this function to the two parameters and uses the output as the new key.

A pre-deployment initialization includes loading the entire set of sensor nodess with the first parameter of each KGF, namely S_b and S_c. The reason for using two parameters is to enable the network to generate a different set of keys for different deployments, so that a node that belongs to the same batch that was not deployed at the same site could not be used by an adversary to launch an attack. The common state preloaded to each node as well as the two seeds are known to the BS prior to deployment. It is not required that the BS knows the unique IDs of the nodes before deployment. However, given a node ID, the BS can determine whether it is valid or not as well as whether the node is a sensor node or a potential CL. Appendix A summarizes the notations and keys used.

5.3.1.2. POST-DEPLOYMENT INITIALIZATION

After deployment, the network starts a top-down bootstrapping process beginning at the base station and proceeding downwards to the cluster leaders and sensor nodes. As shown in Fig. 35, the DCK initialization can be divided into 3 phases. In the following, we discuss the details of the above network bootstrapping procedure.

Phase I: Initializing cluster leaders

After deployment, both the base station and all nodes generate an initial network-wide session key K_{sb} using $KGF_b(S_b,0)$. All cluster leaders use this key to register with the base station by sending their IDs and location encrypted with K_{sb}. Upon receiving such messages, the base station registers all cluster leaders and determines the number of valid *CLs* and accordingly computes suitable value of k_b and m_b. In case of a highly hostile environment with cluster leaders subject to successive capture, the base station may also tabulate the IDs of the cluster leaders with the key combinations and apply a key assignment algorithm such as [122] to ensure that capturing neighboring nodes will not reveal the system keys. Another option that applies to less hostile environments is that every cluster leader uses its ID to hash into the EBS_b key assignment matrix to determine the column that contains its key-string as in [29].

The base station broadcasts to the cluster leaders a new session key K_0^b, k_b, and m_b encrypted with K_{sb} and possibly the key string assignment of each cluster leader. K_0^b is used as the second parameter to KGF_b. Upon receiving this message, each node generates k_b+m_b keys using $KGF_b(S_b,K_b^0)$. Note that sensor nodes at this moment are also capable of generating the k_b+m_b administrative keys.

After knowing its key combination string, each cluster leader purges m_b keys and keeps k_b keys. At this point, the EBS_b system is constructed with every cluster leader knowing k_b keys and every non-cluster leader node knows k_b+m_b keys.

Phase II: Establishing and distributing backup keys

In the second initialization phase, the base station generates a cluster initialization key K_{sc}^i for each cluster leader CL_i , $0<i\leq N_b$, as well as a sequence of shared keys with each cluster's sensor nodes. These keys are unknown to the cluster leader. In order to generate a key sequence,$\{K_i^j\}$, $0 < j \leq t$, of t keys for cluster i, the base station applies a simple hash function to selects K^*_i, one of the m_b keys unknown to CL_i. Then, it uses K^0_i as a seed to a two parameters one-way function, F^i. The base station uses a special random seed, S^*_i, for each cluster as the first parameter to F and K^0_i as the second parameter. The base station generates a sequence of t keys K^j_i, $0<j\leq t$, where, $K^j_i=F(S^*_i, K^{j-1}_i)$. Each key K^j_i, is encrypted with its previous key, K^{j-1}_i and stored to be used as a shared key with the

cluster sensor nodes. It should be noted that once t nodes are evicted from a cluster, the CL should renew its key sequence from the BS.

To initialize each cluster, the base station forms a message for each cluster leader CL_i, with two parts. The first part contains the initial cluster session key, K_{sc}^{i}, which is used for communication between the cluster head and the cluster nodes, and the sequence of t communication keys, each one encrypted with its predecessor. The second part contains both K_{sc}^{i} and CL_i key string, both encrypted with K_i^*, which is unknown to CL_i. The base station then encrypts the entire message with k_b keys known to CL_i and broadcasts the message.

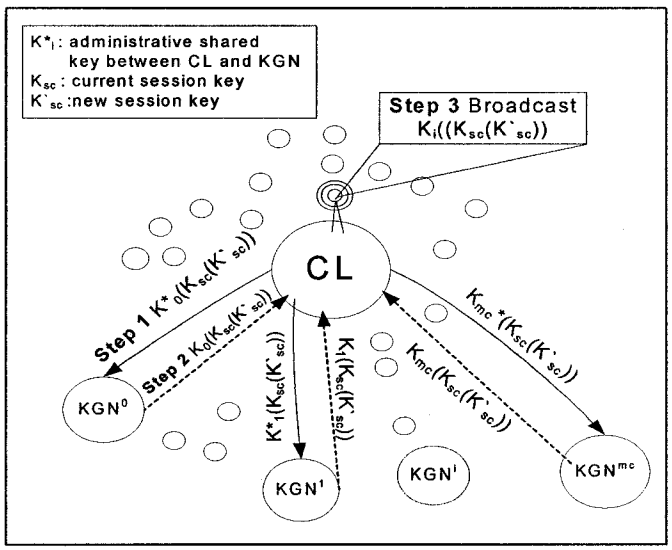

Fig. 36. Refreshing the cluster session key, K_{sc}.

Step 1: CL sends the new session key, $K`_{sc}$, encrypted with the old session key, K_{sc} to m_c KGNs of all keys unknown to CL, encrypted with K*, a shared administrative key between CL and KGN
Step 2: Each KGN decrypts the K* encryption, and re-encrypts with its key, and send back to the CL

After receiving the initial message from the base station, each cluster leader decrypts the message and gets K_{sc}^{i}, then it broadcasts part 2 of message to its cluster sensor nodes

augmented with its key string. Since sensor nodes still keep k_b+m_b keys, each sensor node decrypts the message using K_i* to get K_{sc}^i. It then uses K_{sc}^i to encrypt its ID (and location) and broadcasts to the cluster leader. In case location information is necessary, once they used K_i* to announce their locations, each sensor node purges the k_b+m_b-1 keys used for cluster leaders and keeps the key K_i*.

Phase III: Initializing sensor nodes

The next stage proceeds between cluster leaders and sensor nodes within the cluster is to establish EBS_c the same way as the base station and cluster leaders established EBS_b.. The cluster leader generates a random cluster seed K_c^i, According to the number of sensor nodes in the cluster, N_c, the cluster leader determines suitable values for k_c and m_c and as a key assignment node KAN of the cluster, it generates the $EBS_{ci}(N_{ci},k_{ci},m_{ci})$ assignment matrix. Since sensor nodes within the same cluster are more prone to multiple node capture (to foster collusion between them), DCK applies the collusion minimization key assignment algorithm in SHELL [27].

The cluster leader includes itself a group member and assigns itself a cluster key string. The cluster leader also appoints to each key K_{ci}^j in EBS_{ci} , $0<j\leq k_{ci}+m_{ci}$ a key generation node KGN_c^i. The key generation node is a (sensor) node in the cluster such that, KGN_c^i is already assigned to know K_c^i and shares the maximum possible number of keys with the cluster leader. The cluster leader forms a message containing the key assignments (i.e. key strings and node Ids), the KGN assignments, and the cluster seed K_c^i. This message is encrypted with session key, K_{sc}^i, and broadcasted to all the cluster nodes as well as the base station.

When receiving the above message, each sensor node within the cluster, including the cluster leader CL_i, uses K_c^i to generate k_c+m_c keys using $KGF_c(S_b, K_c^i)$. According to the nodes' key string, each node keeps k_c keys and purges m_c keys in a way similar to [29]. Each node should know the key generators of its k_c known keys. If the node is not a key generator node (KGN) of any key, it purges the entire EBS matrix except its own key string (column). Each KGN_i keeps the row of the EBS matrix that corresponds to the key it is generating and purges the remaining k_c+m_c-1 rows.

To ensure no overlapping takes place between the backup keys nodes communication keys and the administrative keys used to manage the cluster leaders, the cluster head

encrypts the next base station communication key, K_i^j with all k_c+m_c keys and broadcasts it to the cluster. After getting the message, each node in the cluster, except the cluster head, can obtain the new key by decrypting the message both with its administrative keys as well as K_i^*. Each node then purges the old key $K_i^*$to prevent an adversary from knowing it by capturing the node.

At this point of time, the network initialization phase is complete, and the network can start its normal operation. Appendix A summarizes the messages exchanged as well as information stored in different nodes during initialization. At the end of the network bootstrapping phase, the status of the network is as follows:

- Cluster leaders are in a group with $EBS_b(N_b,k_b,m_b)$ key management scheme led by the base station.
- Sensor nodes in each cluster are in a group with $EBS_c(N_c,k_c,m_c)$ key management scheme.
- Each cluster leader knows k_c keys out of k_c+m_c keys of EBS_c, the same as every other node in the cluster.
- Each one of the k_c+m_c keys in $EBSc(N_c,k_c,m_c)$ is assigned a sensor node as a KGN. Each KGN of a certain key K_i knows the corresponding row of the EBS_c matrix, so that, all KGNs can collectively recover the matrix once the CL is captured.
- Each cluster leader has established a session key with its cluster sensor nodes.

Each sensor node in a cluster knows K_i^0, a backup key shared with the base station that is unknown to the cluster leader

5.3.2. NORMAL NETWORK OPERATION

Key management involves generation, assignment and distribution of session keys as well as administrative keys. DCK is composed of upper level EBS to serve base station-cluster leader communication, as well as lower level EBSs to serve cluster leader-sensor nodes communications. Each EBS is used during network operation to regularly generate and distribute a session shared among members for data encryption. The base station creates and distributes also a session key K_{bs}^j to be used among cluster leaders. Each cluster leader CL_i generates and distributes a cluster session key K_{sc}^j shared among all nodes in the cluster to facilitate the first task.

TABLE 3.

Summary of notations and keys used

Notation	Description
BS	Base station
c_i	Cluster i
CL_i	Current cluster leader node for cluster *i*
KGF_i	Key Generation function of sensor nodes (with two parameters)
KGF_b	Key Generation function of cluster leaders (with two parameters)
KAS_{ci}	Key assignment node of cluster *i* (usually CL_i)
KDS_c	Key distribution node of cluster *i* (usually CL_i)
KGN_j^i	The key generation node of key K_j in cluster *i*
$SN_{i,j}$	Sensor Node *j* in cluster *i*
N	Total number of sensors
N_c^i	Total number of sensors in cluster *i*
N_b	Total number of cluster leader nodes
k_{ci}	Number of keys known to each node in cluster *i*
m_i	Number of keys unknown to each node in cluster *i*
EBS_{ci}	The EBS matrix of cluster *i*
k_b	Number of keys known to each cluster head
m_b	Number of keys unknown to each cluster head
EBS_b	The EBS matrix of the cluster leaders
K_i^j	The j^{th} shared backup key between the base station and sensor nodes of cluster *i*
K_i*	A selected key of the m_b administrative keys unknown CL_i and Known to the all nodes in cluster c_i
K_c^i	Initial seed for cluster *i*
K_i**	A selected key of the k_c+m_c administrative keys known to both CL and KGN_i and unknown to the evicted node
K_{sb}	Initial bootstrapping key for cluster leaders
K_{sc}^i	Initial session key for cluster *c*

Session keys are generated by the CL and distributed using EBS administrative keys. Data exchanged during normal network operation is encrypted with group session keys in every cluster. A new group session key is generated and distributed periodically to maintain current secrecy and to protect against brute-force attacks. After every rekeying, the appropriate session key(s) are updated to maintain backward and forward secrecy. Periodic rekeying involves refreshing the [base station, cluster leaders] session key, $K_{bs,j}$,

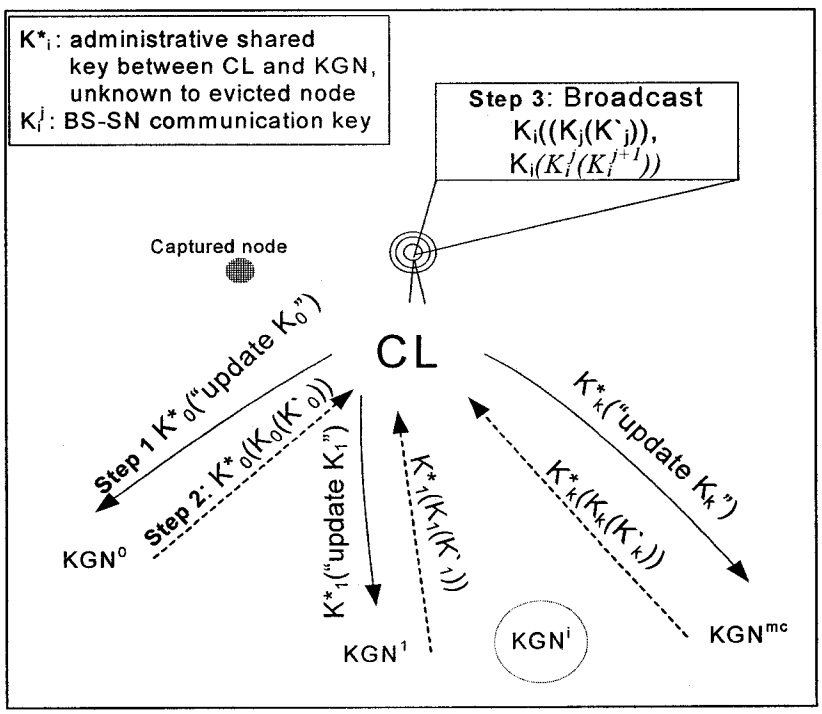

Fig. 37. Handling sensor node capture.

Step 1: CL sends key generation commands to k_c KGNs of all keys unknown to the captured node, encrypted with K*, a shared administrative key between CL and KGN, unknown to captured node
Step 2: Each KGN_i generates $K`_i$ and encrypts it with K_i and further with K* and sends it back to CL
Step 3: CL broadcast $K_i(Kj\ (K`_j))$, for each administrative key K_j unknown to the captured node, in addition to the new BS-SN communication key

and upgrading it to K_{bs}^{j+1}, as well as refreshing each cluster session key $K^{i}_{sc,j}$ and upgrading it to $K^{i}_{cs,j+1}$ for each cluster i, where $0<i\leq N_b$. The new session key is encrypted once with each EBS administrative key and broadcasted to the group. In the following, we discuss periodic rekeying at both levels.

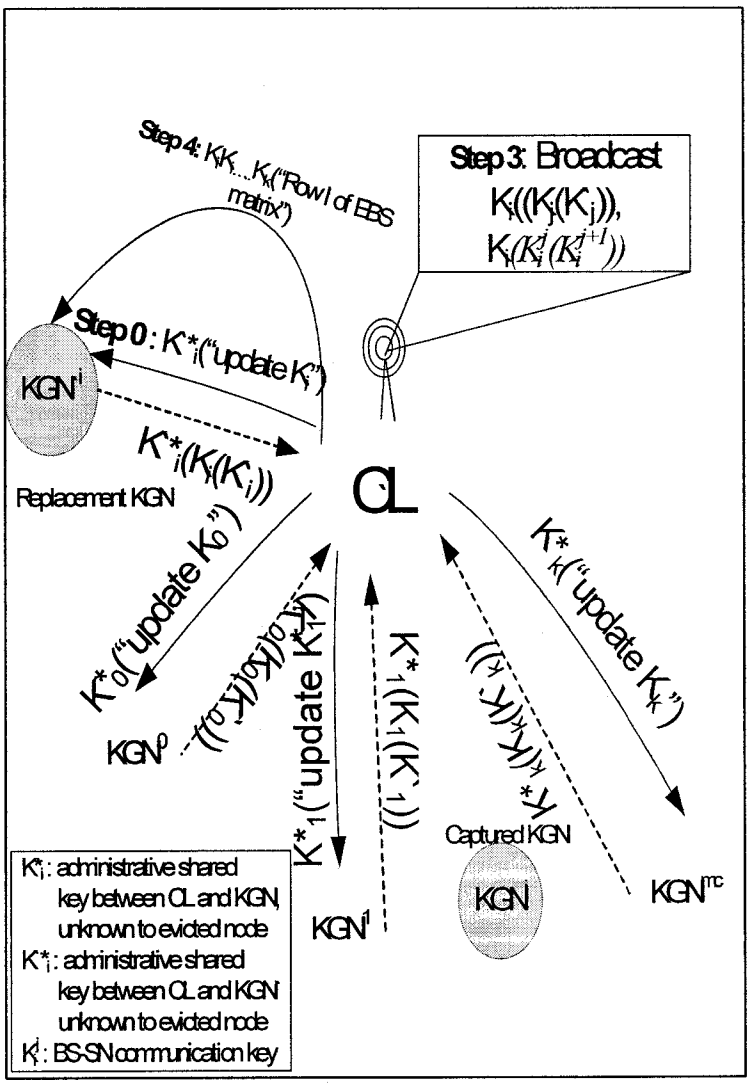

Fig. 38.Ha ndling the capture of a KGN.

Step 0: CL appoints a new KG for K_i using $K`*_I$, a shared key between CL and the new KGN, unknown to the captured KGN and proceeds with **Steps 1,2, and 3** of handling node capture
Steps 2,3: The node eviction procedure proceeds as regular node
Step 4 : CL hands the EBS matrix row to the new KGN

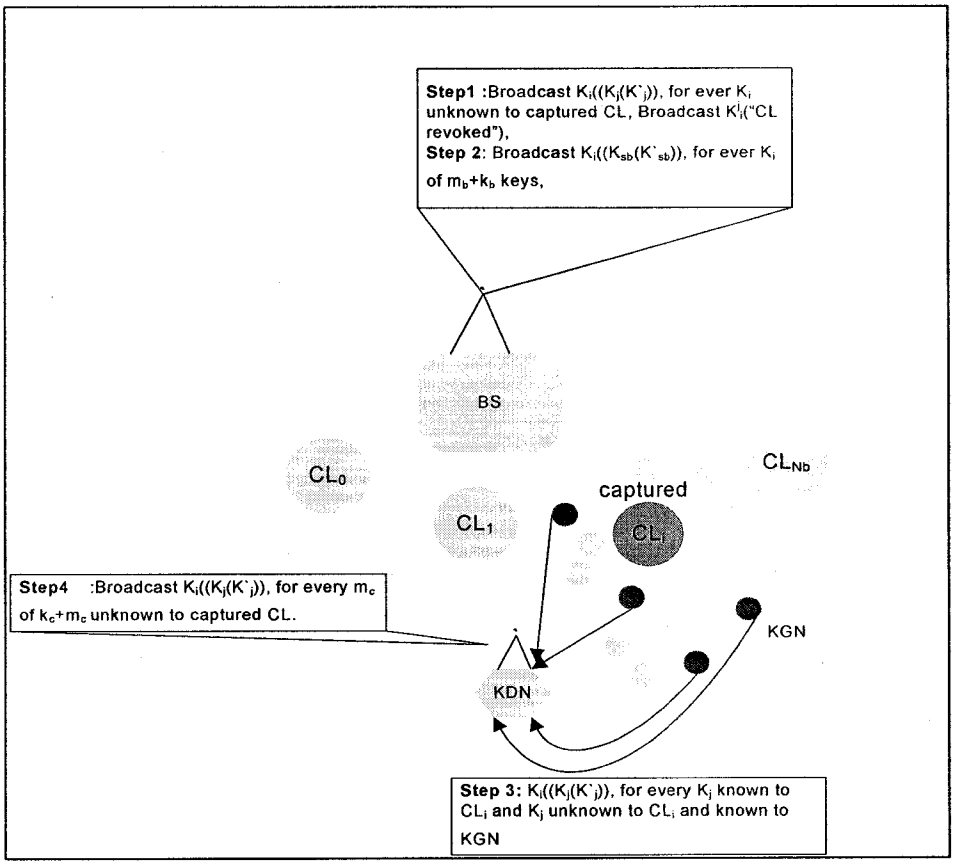

Fig. 39. Handling cluster leader capture, phase 1.

Step 1&2: BS broadcasts new administrative keys to evict CL from EBS_b and a notification to the cluster nodes.
Step 3: each KGN of a key known to the captured CL updates its key and sends it to some KDN
Step 4: KDN broadcasts the new keys to evict CL from EBS_c

Level 1 (EBS_b) Periodic Rekeying In order to refresh $K_{bs,j}$, the base station follows the simple EBS session key refreshing procedure [16]. The base station generates a new session key, $K_{bs,j+1}$, encrypts it with the old session key, $K_{bs,j}$, and then once with every one of EBS_b k_b+m_b administrative keys. The base station broadcasts this message to the

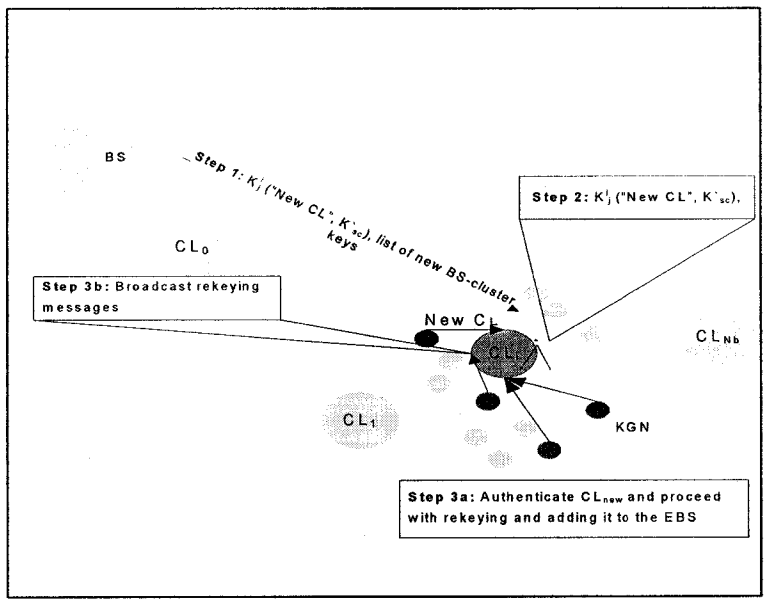

Fig. 40. Handling cluster leader capture, phase 2.

Step 1: BS sends authentication code and all needed status to CL_{new}
Step 2: CL_{new} broadcasts the message to the cluster
Step 3: after authentication, KGNs proceed with normal rekeying to add CL_{new} to EBS_c

cluster leaders. Since each cluster leader knows k_b keys as well as the old session key K_{bsj}, all CLs decrypt the message and update their session key.

Level 0 (EBS_c) Periodic Rekeying Similar to level 1, refreshing the cluster session key K^j_{cij} involves generating the new key and encrypting it once with every administrative key of the EBS_c keys. One feature that distinct DCK from other schemes (e.g. SHELL [122]) is that DCK performs this refreshment in a localized fashion without involving the base station or other cluster leaders. The cluster leader (CL_i) generates a new session key K^{j+1}_{csij}, encrypts it with the previous key as well as once with each of the k_{ci} keys known and broadcasts the message to the cluster. Cluster nodes that know any of the k_b keys known to the CL can decrypt the message and obtain the new key. When the message is received by the KGNs of the m_b keys unknown to the cluster leader, each KGN_i encrypts the message with K_i and sends it back to the cluster leader, which broadcasts the message.

Although this scheme seems to take more time since multiple encryptions take place at multiple nodes, simulation results shoes that it is more efficient than sending the new key to be encrypted in other cluster leaders as in SHELL [122].

5.4. ATTACK MITIGATION

When the network is under attack, a group of nodes might be captured. All the information in the captured nodes is assumed to be revealed. We do not assume nodes to be tamper-proof. However, we assume that some effort is spent in revealing the keys carried by a captured node enough to allow time for the network to apply some attack mitigation mechanism. Recall that a cluster leader is as likely to be captured as any other node while the base station is immune to capture attacks [122]. DCK limits the involvement of the base station in mitigating attacks for the limitations on communications as mentioned earlier. In the following, we describe how DCK mitigates the capture of different types of nodes.

Attack mitigation procedure:

1.-Upon detecting the capture of a sensor node SN_j belonging to cluster i, the cluster leader CL_i determines the key string KS_j assigned to such node.

2.-CL_i issues a revocation message with the key string of the captured node encrypted with the session key and broadc.

3.-Upon receiving such message, each KGN^i_c of a key K_c known to SN_j^i generates a new key K_{c+1} and encrypts it with both the old key K_c and then with all the key K_c, unknown to the evicted SN^i_j. Each KGN^i_c sends the encrypted message to the cluster leader, which broadcasts it to the entire cluster.

4.-After SN^i_j is evicted, it is necessary to change the session key to prevent it from receiving the cluster group traffic. This is done through the session key refreshment procedure described earlier.

5.- To prevent an adversary from obtaining the [base station, sensor node] backup key, K_i^j, the cluster leader distributes the new key encrypted with both the old key K_i^{j-1} and the k_c+m_c administrative keys of EBS_c. **Fig** 37 shows the messages exchanged to implement in the procedure.

5.4.2. KGN CAPTURE

If the captured node plays the role of a key generator of a certain key K_c, it is required to install a new *KGN* to securely replace the captured node. The cluster leader selects a new key generator of K_c according to some policy, among all nodes that know K_c the cluster leader announces this selection to the entire cluster using the cluster leader. The rekeying procedure goes the same as for the general sensor node. After the eviction takes place and the new session key is established, the new KGN needs to obtain its row of the EBS matrix.

The cluster establishes a session key with the KGN and uses it to send this information Fig. 38 depicts the messages exchanged to perform this procedure.

5.4.3. CLUSTER LEADER CAPTURE

Cluster leader capture is one of the most severe attacks on a sensor network. It is usually handled by having the base station either deploy a new cluster leader and reinitialize the cluster or distributing the cluster sensor nodes to join other clusters[14], [122]. The problem with the former approach is that re-initialization involves the loss of all the state information stored in the cluster leader, which might disturb network operation. The latter approach may also disturb network operation by both redistributing the workload and interfering with the new host cluster normal operation.

The objective of DCK is to minimize and localize the key management overhead involved. We assume that the base station will either deploy a new cluster leader or move one of its backup cluster leader nodes to the cluster whose cluster leader was captured. Mitigating such attack involves basically evicting the captured CL from the network, and installing the new replacement *CL*.

In order to revoke the old CL, the base station starts by applying the standard EBS eviction rekeying procedure on EBS_b as follows:

1.-The base station generates new keys for all the k_b keys known to the evicted cluster leader, encrypts them with the old keys and then once with each key of the m_b keys unknown to that cluster and broadcast the message to the entire network.

2.- When the above message is received by other cluster leaders, since each of

TABLE 4.

List of simulation parameters used

Transmission range	150 m
Deployment region	850m x 500m
Simulation time	2×10^4 sec
Frequency of key refresh	5×10^2 sec
Data packet length	10 Kbits
Routing packet length	2 Kbits
Key Size	128 bits
Number of node evictions	10 nodes
Length of shared key chain (t)	10 keys

them knows at least one of the m_b keys unknown to the evicted cluster leader, they can decrypt the message and get the new administrative keys.

3.-The base station then generates a new session key $K_{sb,j+1}$ and the procedure goes exactly as in the refreshing of the cluster leaders session key.

4.-At the cluster c, whose cluster leader was captured, all the sensor nodes receive the above message. Since all sensor nodes know the K_i^j, which is unknown to their evicted cluster leader, they can decrypt the message and know that their cluster leader was evicted. They proceed as follows:

a) Since each sensor node knows the key string of the cluster leader, each KGN of a key K_k known to the cluster leader, generates a new key, encrypts it with the old key and then with all other keys known to it and unknown to the cluster leader.

b) Each one of the above *KGNs* broadcasts the message to the entire cluster (e.g. via controlled flooding).

c) By receiving the above message and decrypting it to get the new keys, the cluster leader is now out of the cluster EBS_c system.

Figure 40 shows the messages exchanged to implement phase 1.

The second step of the procedure is to install a new cluster leader, CL_{new} without re-initializing the cluster. The base station appoints a new cluster leader. The procedure goes as follows:

1.-After rekeying the cluster leaders, the base station adds CL_{new} to the EBS_b using the standard EBS node addition procedure with the same key string as the evicted luster leader.

2.-The base station loads the new CL with k_c keys of the cluster.

3.-The base station encrypts a specific authentication code along with a session key with K_i^j unknown to CL_{new} and gives the session key and the encrypted message to CL_{new}.

4.-CL_{new} broadcasts the encrypted part of the message to the cluster.

5.-After receiving and decrypting the message, each node decide whether or not to verify the new cluster leader.

6.-Once verified, the CL_{new} issues a rekeying command as in section 5.5.

7.-KGNs should proceed with the regular rekeying normally if they have verified the new cluster leader, otherwise, they should refrain.

8.-If the rekeying goes normally, CL_{new} reports success to the base station.

9.-To reestablish the EBS_c matrix, each *KGN* can proceed by announcing its ID, the key it generates as well as its row of the matrix using the session key. Fig 38 shows the messages exchanged to implement this phase.

5.4.4. KEY SECRECY

As stated earlier, once the adversary captures a node, they may uncover all the memory content of such node including administrative keys. The core functional principle of dynamic keying schemes is to invalidate any keys believed to be revealed to attackers so that they are as good as random data. In the following, we analyze how this principle is applied when nodes are captured. The figure gives a brief summary.

5.4.4.1. SENSOR NODE CAPTURE

Each sensor node carries k_{ci} keys of EBS_{ci}, current cluster session key, and the chain of the [base station, senor node] backup keys K_i^j. Cluster rekeying takes place by

generating new k_{ci} keys, encrypting them with the m_{ci} keys unknown to the captured node and distribute them to other nodes. Once the cluster is rekeyed, the cluster leader uses the (currently unknown to the captured node) $k_{ci}+m_{ci}$ keys to distribute the next key in the [base station, sensor node] key chain, K_i^{j+1} encrypted with K_i^{j+1}, which prevents the captured node from obtaining the key.

5.4.4.2. KGN CAPTURE

In addition to all keys held by a sensor node, a KGN of one or more of EBS_{ci} keys knows one row of the EBS_{ci} matrix. Since KGN_i also knows K_i, this information basically tells the attacker, who else knows the same key. Attacking such node will give the attacker the same key, which does not add new keys to the attacker's knowledge.

5.4.4.3. CLUSTER LEADER CAPTURE

A cluster leader is a member in both EBS_b and EBS_{ci}. It knows k keys from each EBS. In addition to these keys, CL knows the key assignment of the cluster as well as the chin of keys shared with the base station. Rekeying takes place at both levels to void the known k_b+k_{ci} keys. The session keys are updated after rekeying, which invalidates the attacker knowledge of such keys. The base station key chain is stored within the cluster leader in an encrypted form. New chain of keys is delivered to the new cluster leader encrypted with the last key known to the sensor nodes. Finally, the EBS_{ci} is recoverable to the new CL through the KGNs. Tables 5 shows the security analysis of both sensor node and cluster leader captures.

5.5. USING KEY POLYNOMIALS WITH DCK

In order to reduce the number of keys known to the attacker by capturing nodes, we propose the use of key polynomials instead of traditional keys. Each node is assigned k polynomials out of existing $k+m$ polynomials. Key polynomials are used in [8] and [70] in conjunction with key pre-distribution to enhance resiliency to node capture. The scheme in [70] does not employ rekeying. We propose to use key polynomials in an EBS-based scheme to make use of efficient rekeying to further enhance resilience to node capture and consequently enhance network survivability.

In our proposed scheme, similar to Liu and Ning's [70] are pre-distributed to nodes. Two nodes can establish a pair-wise key if they share a key polynomial. A group session key can be encrypted $k+m$ times, each time with a key derived by evaluating F_i, the i^{th} polynomial at the base-station. Each node can decrypt the session key with at least one of the polynomials known to it.

Rekeying to evict a node can be done by generating k new replacement key polynomials. The new polynomials are encrypted m times each time with a key derived by evaluating the i^{th} polynomial, $0 \leq i < m$, of the m polynomials unknown to the evicted node(s) and broadcasted to the sensors.

The major advantage that can be achieved by our scheme is the reduction of the attacker's ability to obtain key polynomials by capturing nodes since he needs at least $t+1$ shares of each polynomial. This is studied through simulation in chapter 6.

In EBS-based schemes that use traditional keys, each node has to store k keys (typically 128 bits each) in addition to a bit-string of length $k+m$ bits. In basic key pre-distribution[30], each node also stores k keys. In key polynomials pre-distribution [70] *t-degree* key polynomials are established over a finite field F_q, and each node stores the shares of k polynomials. Accordingly, each node stores $k*(t+1)logq$ bits. Our scheme incurs similar storage overhead with $k(t+1)logq$ bits in addition to a $k+m$ bits key string.

Since static schemes do not alter administrative keys, communication overhead for administrative rekeying is not applicable. In previous EBS-based schemes, each rekeying involves m messages of length $k*key\_size$ bits each. In our scheme, each rekeying involves the distribution of m messages each with $(t+1)logq$ bits.

TABLE 5.

Protecting network keys after node capture

Node Type	Keys	Mitigation
sensor node	EBS_{ci} k_{ci} keys	The cluster leader performs EBS rekeying at EBS_{ci} using the remaining m_{ci} keys
	K_{sci}	The cluster leader refreshes the session key of the EBS_{ci} right after revoking SN_i
	K^i_j	After rekeying, CL replaces K^j_i with K^{j+1}_i encrypted with EBS_{ci} m_{ci} unknown to SN_i as well as K^i_j. SN_i needs both keys to obtain K^{j+1}_i, which is not the case.
	EBS_{ci} matrix row (in case of KGN)	The matrix is recoverable through the *CL*
	EBS_b k_b keys	The base station performs EBS rekeying at EBS_b using the remaining m_b keys
	K_{sb}	The base station refreshes the session key of the EBS_b right after revoking CL_i
cluster leader	EBS_{ci} k_{ci} keys	Receiving the base station notice, each of the k_{ci} KGN^i_j generates a new key K_j`, encrypts it with the old key K_j and then with all EBS_{ci} keys unknown to CL_i and broadcasts it to the cluster
	K^i_j	CL_i knows only an encryption of K^j_i with the previous unknown key. A new replacement CL is equipped with another set of keys.
	K_{sc}	Once rekeying takes place, K_{sc} is obsolete.
	EBS_{ci} matrix	The matrix is recoverable through the *KGNs*

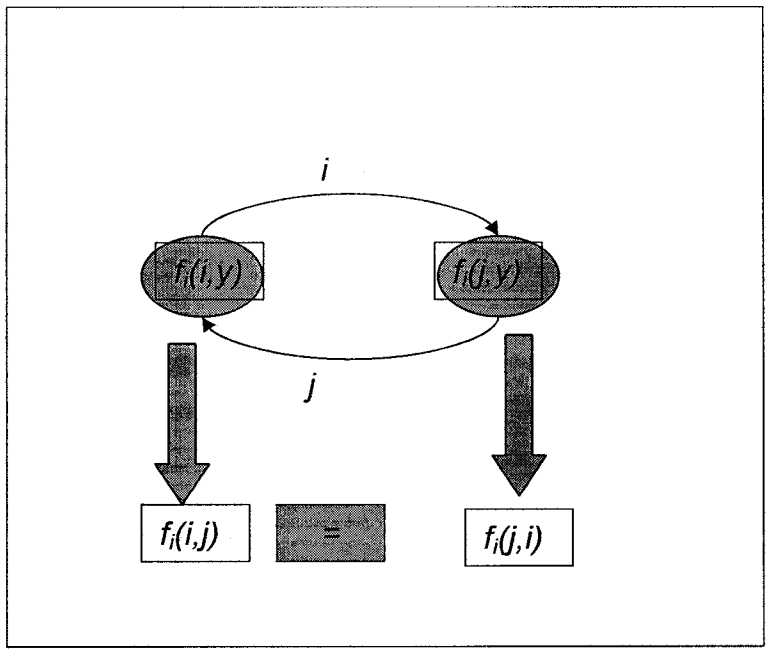

Fig. 41. Establishing pair wise keys.

The following is a summary of using key polynomials for establishing pair wise keys, establishing group keys and rekeying.

5.5.1. ESTABLISHING PAIR WISE KEY

Currently, plain cryptographic keys (e.g. 128 bits key) are used. Once a node is captured, all keys known to that node are considered compromised. t-degree bi-variate polynomials, f(x,y), may be used instead of plain keys. Accordingly, f(x,y)=f(y,x). Each key Ki is replaced with a polynomial f^i. Regular keys can be considered key polynomials with t=0. Node Ni receives a share uni-variate polynomial, fj(i,y), for each key Kj,,1≤j≤k. Two nodes can establish a pair-wise key if they share a polynomial.

5.5.2. ESTABLISHING GROUP KEY

Session key encrypted $k+m$ times, each time with a key derived by evaluating *fi*, the *ith* polynomial at a certain point known to all nodes. Each node can decrypt the session key with at least one of the polynomials known to it. Rekeying to evict a node

can be done by generating k new key polynomials. New polynomials are encrypted m times each time with a key derived by evaluating the *ith* polynomial, $0 \leq i < m$, and broadcast to the sensors. For an attacker, in order to know a polynomial, he needs to capture at least $t+1$ nodes.

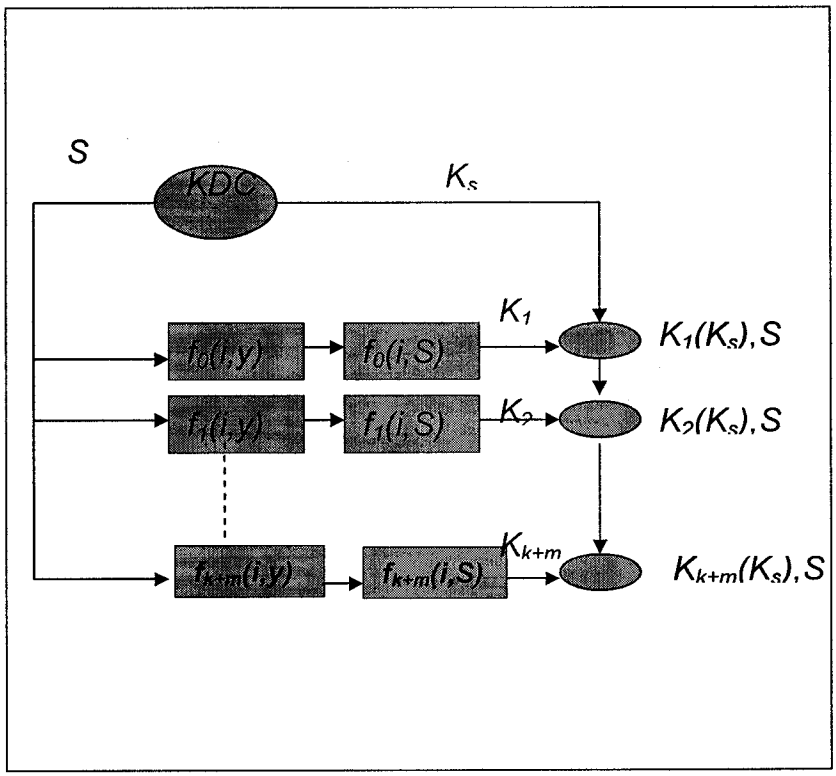

Fig. 42. Establishing Group Session Key.

5.5.3 REKEYING

New administrative key polynomials are generated. New polynomials are encrypted m times each time with a key derived by evaluating the i^{th} polynomial, $0 \leq i < m$, and broadcast to the sensors. Each node can decrypt the administrative key if it has the key with at least one of the polynomials known to it. For an attacker, in order to know a polynomial, he needs to capture at least $t+1$ nodes.

5.6. CONCLUSION

In this chapter we proposed a solution for WANs. We started with comparing static and dynamitic keying schemes. Static key management schemes tend to enhance network resilience to node capture at the cost of lower network connectivity. Dynamic key management schemes preserves connectivity and handles node capture through rekeying using a centralized KDC. Our proposed scheme enhances network resilience by using key polynomials similar to static schemes. We handle node capture through rekeying similar to dynamic schemes.

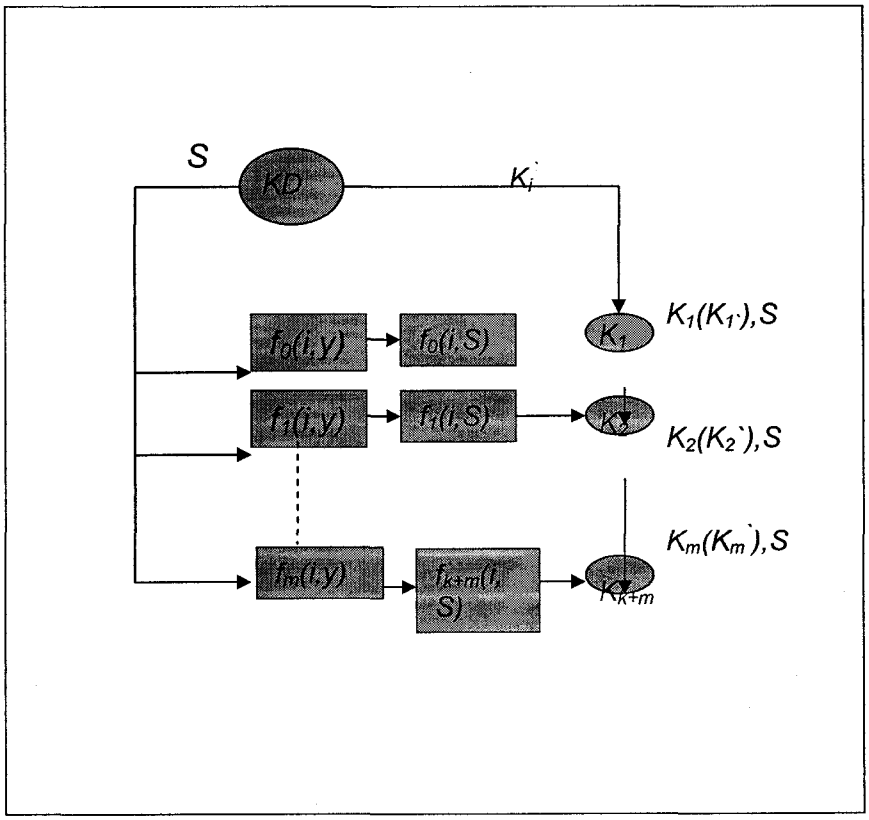

Fig. 43. Rekeying under key polynomials.

We proposed Dynamic Combinatorial Keying (DCK) which combines key analysis, generation, assignment and distribution. Since sensor networks are subject to more attacks, we proposed the used of key polynomials to make capturing the network even harder. In the next chapter, we analyze the proposed solution and compare it to existing solutions through simulation.

CHAPTER VI

SCHEME RESULTS IN WIRELESS SENSOR NETWORKS

In this chapter, we review the simulation results of the two parts of our proposed scheme for Wireless Sensor Networks. In section 2, we view the simulation results of DCK. In section 3, we overview the simulation results of the key polynomial. In section 4, we focus on the possible extensions and future work.

6.1. DCK PERFORMANCE EVALUATION

Security solutions for sensor networks usually involve a tradeoff between the required security level and the resource consumption. In order to evaluate the resource consumption in DCK, we performed a simulation study that compare DCK to another EBS-based key management scheme, SHELL [122]. Since the authors in [122] showed their scheme to consume less energy than *Jolly's* scheme for dynamic keying [121], we consider only SHELL [122] for comparison. We simulated networks of different populations, namely, 500 to 950 nodes distributed in 10 clusters in a deployment field of 850 x 500 meters. We ran our simulation for 20,000 seconds. We consider the cost of key management in terms of energy consumed on average by a sensor node and a cluster leader. This cost can be divided into setup cost, key refreshment cost and node eviction

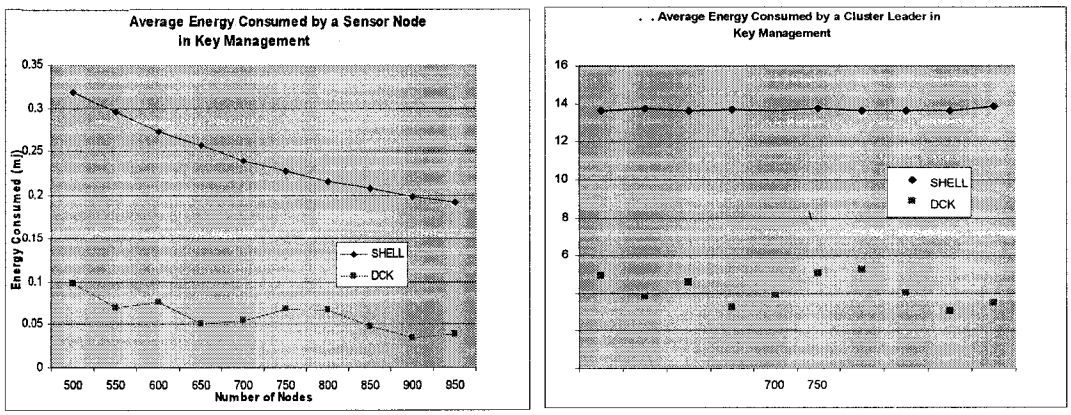

Fig. 44.Av erage energy consumed by a sensor node.

cost. Results show that the cost of key management in both schemes is mostly incurred on the cluster leader nodes rather than the senor nodes.

6.1.1. ENERGY CONSUMPTION IN KEY MANAGEMENT

To measure the overhead of key management, we ran our simulation without key management and then with each of DCK and SHELL and measured the difference in energy consumption as an indicator of the key management cost.

Fig. 44 shows the average energy consumed by a sensor node and a cluster leader in both solutions for key management. It can be seen from the figure that using DCK, a sensor node on average consumed one order of magnitude less energy than SHELL. The average energy consumed by a cluster leader in our approach is less than half of the energy used by a gateway in SHELL. This might be due to the fact that in SHELL, a sensor node shares a session key with two cluster leaders other than its own as well as a unique key with its own cluster leader, which is used to form an individual message per senor node. On the other hand, in DCK, we localize key refreshment through the use of KGNs within the same cluster. In order to factorize the energy used in key management, we measured the energy used for setup, regular key refreshment and the random evictions. It can be seen from Fig. 46 that the major portion of the key management energy is consumed in periodic key refreshment. Since our approach localizes regular key refreshment within the cluster, DCK saves more than two thirds of the key refreshment energy.

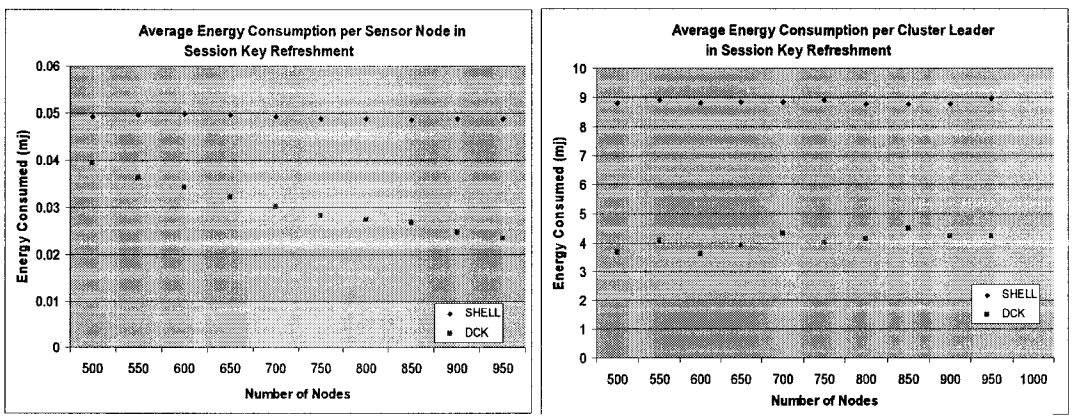

Fig. 45. Average energy consumed by a sensor node in session key refreshment.

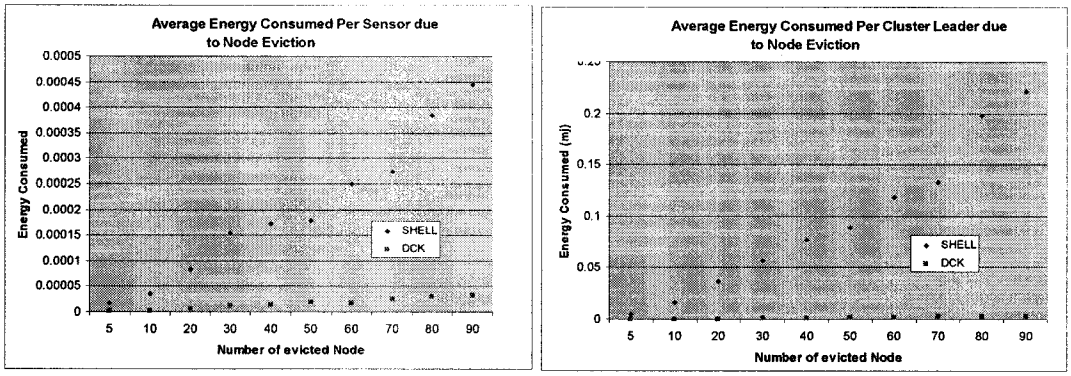

Fig. 46. Average energy consumed by sensor node due to eviction.

6.1.2. ENERGY CONSUMPTION IN NODE EVICTION

One of the major advantages of DCK is the handling most node eviction locally within the cluster. In Fig. 46, we show the average energy consumed by a sensor node and a cluster leader in evicting a number of nodes that varies from 5 to 90 nodes in a 750 node network. It is expected for the energy consumed to grow linearly with the number of evicted nodes which is shown to happen in both schemes in Fig. 46. Since our approach handles node eviction locally within the cluster without involving other cluster leaders or the base station, we consume much less energy than the scheme in [122]. It can also be noticed that in SHELL, the major portion of the node eviction energy is consumed by the cluster leaders. On the other hand, the major portion of total energy is consumed by sensor nodes. A possible explanation is that according to [122], in order to evict a node, the cluster leader should contact other cluster leaders assigned as key generators to generate the new keys and send them back to be distributed to the cluster. This inter-cluster dependency mandates that evicting a sensor node or a cluster leader in a certain cluster affects other clusters (e.g. at least two other key generator clusters [122]). In our approach, the key generation responsibility is distributed locally among KGNs within the same cluster, which saves energy and localizes the effect. Hence, node eviction does not have a global effect on the network, which contributes in scalability.

6.1.3. STORAGE AND SCALABILITY

In SHELL, the base station establishes inter-cluster keys between each two clusters and then selects for each cluster d (typically 2) other key generation clusters to share keys with the original cluster's nodes.

Accordingly, cluster leaders (a.k.a. gateways) in SHELL can be viewed to form a fully connected communication graph with respect to the inter-cluster keys, and a directed graph with an out degree of exactly d with respect to the key generation relationship. Accordingly, evicting a cluster leader involves the updates of n inter-cluster keys as well as the update of d sets of keys shared between the key generation clusters and the sensor nodes of the original cluster. In our approach, we replace the $n(n-1)/2$ inter-cluster keys with the EBS_b matrix which scales better in the number of clusters. Each cluster leader in DCK stores a chain of t (typically 10) backup keys shared between the sensor nodes and the base station. Fig. 47 shows the number of keys maintained by a cluster leader against network sizes in both schemes. Since both schemes use EBS to manage sensor nodes within each cluster, the number of administrative keys maintained by a sensor node is

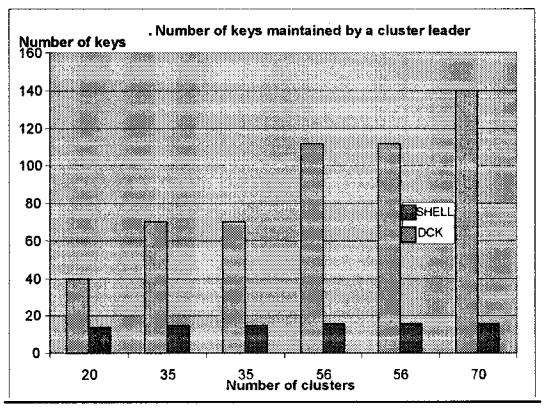

Fig. 47. Number of keys managed by a cluster leader.

about the same. In SHELL, each sensor node shares an individual key with the base station, whereas, in DCK the same backup key is shared between the base station and all sensor nodes in a specific cluster. Sensor nodes in a specific cluster in SHELL share also

a key with the key generation gateways, which is not the case in DCK. Hence, number of keys held by a sensor node is almost the same in both schemes.

6.2. KEY POLYNOMIALS PERFORMANCE EVALUATION

In this section, we compare our proposed key polynomials scheme to leading existing static and dynamic keying schemes. The major points of interest are (1) the network resilience, measured as the number of captured nodes required for the attacker to control the network; (2) the network connectivity, measured as the probability for any two randomly selected nodes to communicate directly; and (3) the rekeying overhead.

6.2.1. COMPARISON WITH STATIC KEYING

As described in section 2, static keying schemes depend on the pre-distribution of a randomly selected set of k keys [35] or bi-variate key polynomials [70] to each node out of a pool of P=k+m of keys. Two nodes can communicate directly if they are within the transmission range of each other and they share a key [70] or a key polynomial [35]. Since the key polynomial model is a generalization of the basic probabilistic model, we focus on the former without loss of generality.

The probability of sharing a key polynomial (and consequently establishing a pair wise key) P_s, is defined as follows:

$$P_s = 1 \qquad \text{if } k>m,$$

$$P_s = \left(1 - \prod_{i=1}^{k-1} \frac{m-i}{k+m-i}\right) \quad \text{if } k \leq m, \tag{21}$$

Where k is the number of key polynomials known to each node and m is the number of polynomials unknown to each node.

If two nodes are not in direct communication with each other, they use a third-party that shares a key with each one as an intermediary. The network connectivity is defined as the probability of direct communication between any two nodes. The network topology as well as the transmission range determines the physical network connectivity, whereas the probability P_s is an indication for logical connectivity.

Static keying schemes tends to select a large value of k (e.g. 250 or more [35]) to guarantee a suitable connectivity (e.g. P_s=0.99). However, since the value of k is directly

related to the node storage capacity C, it might not be a design option to use such large value of k, especially in case of polynomials where $C=(t+1)k$, where t is the polynomial degree. On the other hand, in order to enhance the network resilience to node capture, m is selected to be quite large so that more nodes are needed to reveal all keys. However, the larger value of m the less connected the network is since the probability of sharing a key is lowered. The relationship between resilience, connectivity, and m in a 20,000 network is shown in **Fig.** 48.

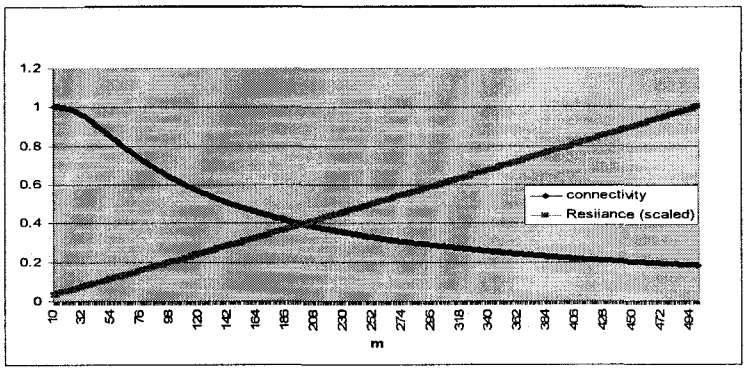

Fig. 48. Netwok connectivity and resiliance against m.

Since our proposed scheme uses the same idea of selecting k polynomials for each node as in [70], formula (21) holds. However, our model operates at the lower values of m for the connectivity and thus guarantees high connectivity. Resilience is achieved through rekeying since new keys are generated and distributed regularly to compensate for keys revealed by node capture.

The probability of compromising a specific key polynomial, Fi, given the capture of N_c nodes given that they communicate directly is given by (22)

$$p_{F_i} = 1 - \sum_{i=0}^{t} \binom{N_c}{i} \left(\frac{k}{k+m}\right)^{i} \left(\frac{m}{k+m}\right)^{N_c - i} \qquad (22)$$

The probability of any direct or indirect key to be compromised is

$$P_c=P_s * p_{Fi}+(1-P_s)((1-p_c)\ (1-p_F)^2), \quad (23)$$

Where $p_c=N_c/N$.

In [70], these probabilities were studied against the polynomial degree t and it was shown that the network resilience is maintained up to a certain threshold N_c^* of captured nodes, with the probability very close to zero. When N_c exceeds N_c^*, the probability rises rapidly very close to one. When Pc is close to one, this means that capturing a certain number of nodes will reveal all the networks keys, which means in other words, the network was fully captured and controlled by the attacker. Fig. 49 shows the network capture probability against the number of captured nodes N_c. We define the network capture point N_c^* as the number of captured node at which the capture probability is very close to one. At this point, the number of keys revealed to the attacker is $k+m$, which indicates that the attacker has full control of the network. N_c^* is considered a measure for network resilience.

Different parameters affect the value of N_c^* including k, m, and t. In regular EBS, using traditional keys can be considered as polynomials of degree zero. It can be noticed that increasing the polynomial degree t enhances the network resilience on the expense of storage per node. It can also be noticed in using regular keys, capturing very few number of nodes is enough to capture the network as long as the attacker is assumed to foster collusion among these nodes that are not necessarily co-located. Although enhances connectivity increasing k reduces the value of Nc^* as shown in Fig. **50** 48. Increasing m enhances resilience but significantly reduces connectivity.

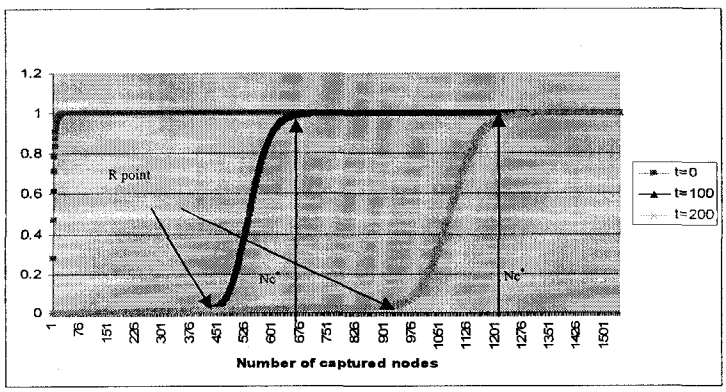

Fig. 49. Probability of network capture.

In our approach, we try to perform rekeying early enough before the number of captured nodes reaches N_c^*. Since the node capture probability rises rapidly before N_c^*, we can perform rekeying on the rising edge of the node capture probability at a point we call R-point. The rekeying overhead is sending m messages, each contains k keys. One obvious way to minimize the number of times rekeying is needed is to enhance the network resilience (N_C^*), by using a lower value of m and a quite larger value of k. Since rekeying is performed early enough before reaching N_c, we can afford to use a smaller value of t to save the storage space without jeopardizing the network security. This is shown in the next sub-section with dynamic keying schemes.

6.2.2. COMPARISON WITH DYNAMIC KEYING

In dynamic keying schemes, rekeying is performed regularly (before the number of captured nodes reaches N_c^*. In order to evaluate the network resilience to attack, dynamic key management schemes consider the number of keys revealed to the attacker by capturing N_c nodes as the measurement of network resilience to attack. This is depicted in Figure 50.

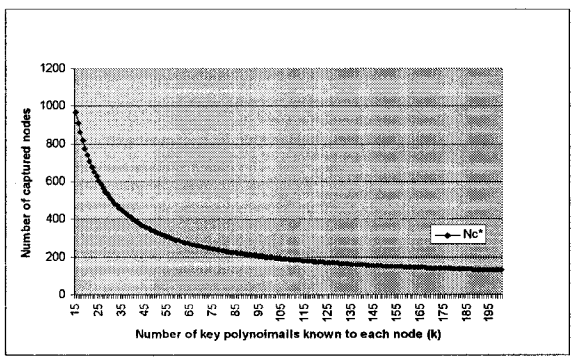

Fig. 50. Number of nodes needed to control the network.

We simulate our scheme in the same environment used in location-based key assignment [124]. For simulation feasibility, our network consists of 40 to 200 nodes deployed in a 500x500 field. We assume that the attacker succeeds in capturing a number of nodes that varies between 5 to 55 nodes. The transmission range is 55 meters. We select different values of the EBS parameters k and m that totals to 10. To lower the storage overhead, we use *t=0, 5, and 10.* We study the effect of using key polynomials on both random and location-based key assignment [124]. Traditional EBS keys can be considered using constant polynomials, i.e. *t=0.*

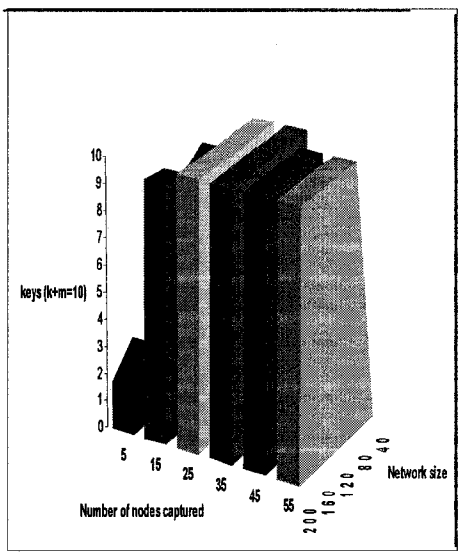

Fig. 51. Average number of keys revealed under random key assignment with *t=0.*

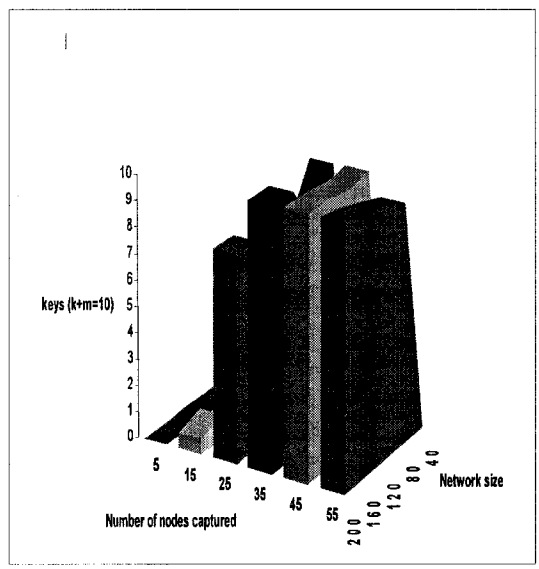

Fig. 52. Average number of keys revealed under polynomial random key assignment with *t=5.*

In Fig. 51, Fig. 52, and Fig. 53, we plot the average number of keys revealed under random key assignment for *t=0,5, and 10 respectively*. In Fig. 54, Fig. 55, and Fig. 56, we plot the average number of keys revealed when under location-based key assignment [49], for *t=0,5,10* respectively.

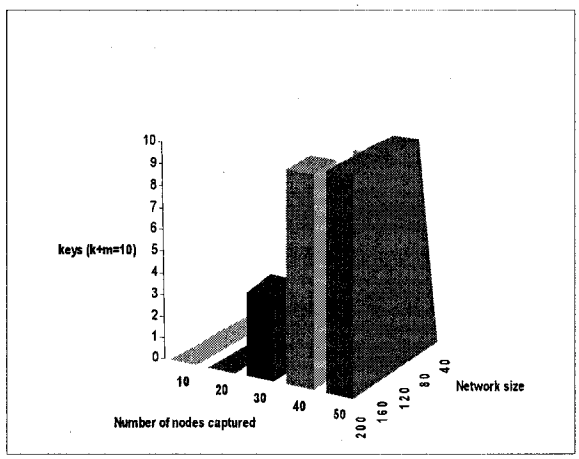

Fig. 53.Ave rage number of keys revealed under polynomial random key assignment with *t=10*.

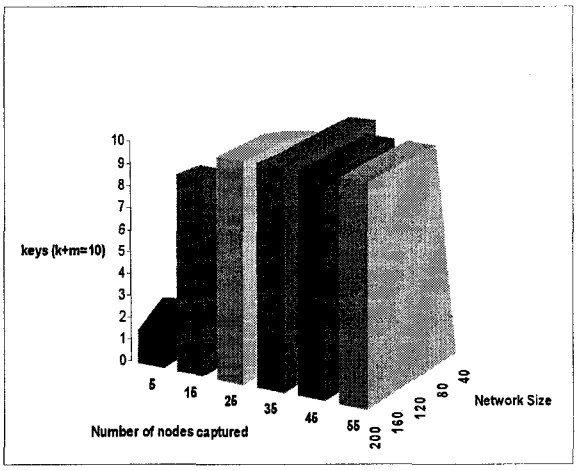

Fig. 54. Average number of keys revealed under location-based key assignment with $t=0$.

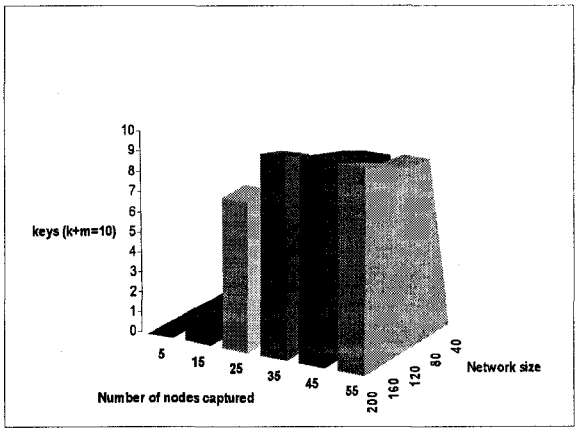

Fig. 55. Average number of keys revealed under polynomial location-based key assignment with $t=5$.

It can be seen from Fig. 51 and Fig. 54 that location-based key assignment is better than random key assignment in small number of captured nodes and network sizes. In case of large number of captured nodes and large network sizes, location-based key assignment might not help much since the probability of captured nodes being close to each other and able to exchange keys is high.

In Fig. 52 and Fig. 55, it can be seen that using key polynomials reduces the number of keys revealed significantly since it is needed to know at least *t+1* shares to reveal a single key polynomial. If the number of captured nodes that can collude is less than *t*, no keys are revealed. In case of large number of captured nodes, and/or large network size, the number of revealed keys increases since the number of shares available increases. However, the probability of revealing all the keys is very small since one need to have *t+1* shares for each key. It can also be noticed that using key polynomials in conjunction with location-based assignment reduced the number of keys revealed in small number of captured nodes but not in large number of captured nodes for the same reason. This enhancement comes at the cost of sensor nodes ability to know their locations.

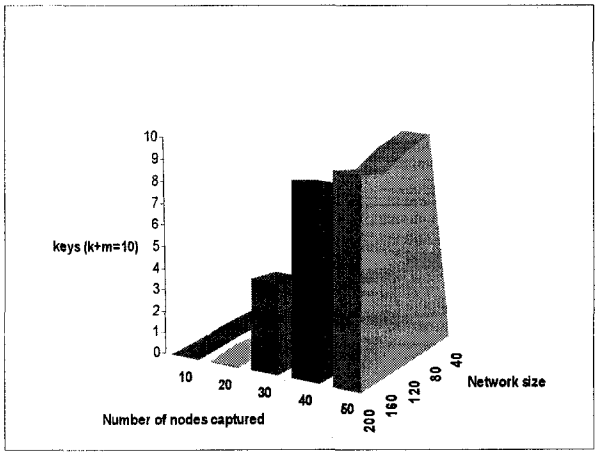

Fig. 56. Average number of keys revealed under polynomial location-based key assignment with *t=10.*

In Fig. 53, and Fig. 56, it is shown that increasing the degree *t* helps in reducing the number of keys revealed further in small number of captured nodes. However, if the number of nodes captured is high, increasing *t* does not help much since the overlap between shares known by captured nodes increases significantly.

In this thesis, we classified key management schemes in sensor networks as either static or dynamic based on whether the administrative keys are updated or not after network deployment. We proposed a new dynamic combinatorial key management scheme based on key polynomials and compared our scheme to existing dynamic and static schemes.

Most existing static keying schemes tend to enhance network resilience to node capture at the cost of lower network connectivity. Dynamic keying schemes preserve connectivity and handle node capture through rekeying. Our proposed scheme enhances network resilience to capture by using key polynomials similar to Liu and Ning's static scheme. We handle node capture through EBS-based key assignment and administrative key rekeying similar to other EBS-based dynamic schemes.

Comparing our schemes to Liu and Ning's, our scheme was found to enhance network resilience to node capture at a smaller polynomial degree t and accordingly with less storage per node. Most notably, our scheme preserves network connectivity by using a smaller key pool. Compared to location-based dynamic key management schemes, our scheme was found to provide significantly higher network resilience to collusion at the low cost of using degree t (typically 5 to 10, compared to 50 to 100 in Liu and Ning's scheme) key polynomials instead of using constant traditional keys.

Finally, we note that this work presented a preliminary comparison of static versus dynamic keying. We are currently conducting a more comprehensive analysis of various schemes. We are also developing a localized multi-tier EBS-based scheme for clustered networks.

6.3. CONCLUSION

In this chapter, we evaluated our solutions for WSNs. We started with DCK and compared it to existing solutions. Unlike most key pre-distribution schemes, DCK uses administrative and communication key refreshment and emphasizes rekeying upon node capture to maintain secure communications. DCK decouples key generation, assignment and distribution, and maintains regular key refreshment as well as handling node captures locally within the cluster. Simulation comparison with the EBS-based SHELL key management scheme show that: (1) cluster leader nodes carry the major part of the key management overhead; (2) DCK consumes less than 50% of the energy consumed by

SHELL in key management; (3) localizing key refreshment and handling node capture enhances the security by minimizing the amount of information known by each node about other portions of the network; and (4) since DCK does not involve the use of other clusters to maintain local cluster data, it scales better from a storage point of view with the network size represented by the number of clusters.

Comparing our key-polynomial based schemes to static key polynomials, our scheme was found to enhance network resilience at a smaller polynomial degree t and accordingly with less storage per node. Our scheme was preserves the network connectivity by using a smaller key pool. Compared to location-based dynamic key management schemes, our scheme was found to provide around significantly better network resilience at the cost of using degree t (typically 5 to 10) key polynomials instead of using constant keys.

CHAPTER VII

CONCLUSION AND FUTURE EXTENSIONS

In this section, we conclude this dissertation by summarizing our motivations, objectives, contributions, and the performance of our proposed Key Management schemes. Furthermore, we discuss a list of possible future extensions to our work in the context of key management in ad-hoc and sensor networks.

7.1. CONCLUSION

In this sub-section, we conclude our proposed key Management schemes for Ad-hoc and Sensor Networks. We start with Ad-hoc networks and then we follow by Sensor networks.

7.1.1. CONCLUSION AND RESEARCH DIRECTIONS FOR AD-HOC NETWORKS

In this thesis, we proposed a novel architecture for secure group communications in WAHNs. We claim that our architecture can readily be populated with components to support objectives such as fault tolerance, full-distribution and scalability to mitigate WAHN constraints. In our architecture, group management is integrated with multicast at the application layer.

We presented a comprehensive group management module, an efficient decentralized scheme that performs key assignment, generation, and distribution in compliance with this architecture. We overviewed each of these components and showed how they meet the architectural objectives. We performed some analysis on the performance parameters of each component separately and showed the feasibility of such design compared with existing architecture. Currently, we are integrating all the key management and group session management components together and define all the interaction. Our next step is to perform simulation experiments on such integrated group management scheme and compare real network parameters, such as actual delays and traffic overhead, with other existing key management schemes.

We started by CKDS, in which we combined two scalable schemes to introduce an efficient application-level key management protocol for secure group communication in wireless ad-hoc networks. We used CAN for application-level multicast and EBS for key distribution. We assumed a centralized group controller that performs key generation and rekeying message construction. We introduced two new key distribution schemes, *m*-dimensional multicast and 2D multicast to deliver keys efficiently to group members in such an environment. We used Directed Flooding as a multicast message forwarding technique in our approach at individual quadrant level. We compared our scheme to using this type of flooding solely to deliver keys.

Our technique achieved a significant network traffic reduction of almost one eighth of it in using normal application-level multicast in CAN. We compared storage requirement and overhead incurred on each node to GKMP. Our scheme showed better scalability and lower storage requirement at the cost of small number of decryptions performed by each individual node. We showed that selecting lower number of keys per user, and accordingly higher number of rekeying messages, in EBS achieves better performance under our 2D multicast key distribution scheme. We showed our approach to be scalable with respect to both storage and communication. We believe this approach is suitable for mid size wireless ad-hoc networks with hundreds or thousands of users.

We presented EDKMS, an efficient decentralized scheme that performs key assignment, generation, and distribution without assuming trusted nodes. EDKMS is based on combinatorial exclusion basis systems, application-level multicast, verifiable secret sharing, and application-based key clustering. EDKMS offers batch re-keying with collusion resistance, threshold-based verifiable key generation, and scalable key distribution. All three functions have been rigorously analyzed. Our future work includes complete protocol specification and prototype implementation. An interesting prospect is extending our work to wireless sensor networks that are characterized by a huge number of sensor nodes each with limited capabilities.

7.1.2. CONCLUSION AND RESEARCH DIRECTIONS FOR SENSOR NETWORKS

Key management in sensor networks raises interesting research issues. Design requirements for key management solutions include energy awareness, survivability and localization of attack impact given a highly vulnerable network that mainly operates unattended, and scalability to a large dynamic network. Administrative key pre-distribution has been the de-facto solution with the underlying assumption that a key will outlive the network. Recently, dynamic schemes (with re-keying) have been proposed particularly for emerging long-lived sensor networks. Efficient re-keying is essential for these schemes to be adopted. This article showed that both static and dynamic keying share some basic concepts. A study of network resilience to attacks as well as storage and communication overhead for both classes of schemes has been presented.

One major challenge to dynamic keying schemes is the need for the participation (to varying degrees) of a key management authority (usually the base station) post network deployment. This participation is not needed in static schemes (as there is no re-keying). Research is ongoing to decrease the reliance on a key management authority in dynamic key management schemes such as LOCK. Other notable leading and promising research efforts on key management in sensor networks include Ning et al at North Carolina State University [71], [72], Perrig et al. at Carnegie Mellon University [4], [28], Zhu et al. at The Pennsylvania State University [108] and Hartel et al. at the University of Twente (EYES project) [63].

We started with DCK, a modified scheme suitable for WSNs. 1) cluster leader nodes carry the major part of the key management overhead; (2) DCK consumes less than 50% of the energy consumed by SHELL in key management; (3) localizing key refreshment and handling node capture enhances the security by minimizing the amount of information known by each node about other portions of the network; and (4) since DCK does not involve the use of other clusters to maintain local cluster data, it scales better from a storage point of view with the network size represented by the number of clusters. Optimizing DCK parameter values for each cluster involves a tradeoff between the cost of evicting a captured node and the cost of deploying new nodes. During the lifetime of a

sensor network, multiple deployments might take place to compensate for damaged, failed, dead or attacked sensor nodes. Currently, we are investigating the dynamic adaptation of the cluster structure during the lifetime of the network to achieve optimal performance. Current and future research also includes using group key polynomials for collusion mitigation. Our preliminary research on using group key polynomials in flat networks showed promising results. We are also conducting an extensive study of static versus dynamic keying schemes.

Static key management schemes tend to enhance network resilience to node capture at the cost of lower network connectivity. Dynamic key management schemes preserves connectivity and handles node capture through rekeying using a centralized KDC. Our proposed scheme enhances network resilience by using key polynomials similar to static schemes. We handle node capture through rekeying similar to dynamic schemes.

Comparing our schemes to static key polynomials, our scheme was found to enhance network resilience at a smaller polynomial degree t and accordingly with less storage per node. Our scheme was preserves the network connectivity by using a smaller key pool. Compared to location-based dynamic key management schemes, our scheme was found to provide around significantly better network resilience at the cost of using degree t (typically 5 to 10) key polynomials instead of using constant keys.

7.2. LIMITATIONS AND POSSIBLE FUTURE EXTENSIONS

Although our solutions contribute to solving several problems, such as multiple node capturing, collusion, energy saving, there are several other problems that are not yet addressed. Our techniques do not present solutions to such problems.

The first problem is excessive mobility. In a sensor or ad-hoc network with nodes moving very fast, maintaining the cluster structure of the network with the existing or newly introduced cluster leaders is very complicated if the mobility exceeds certain limits. Maintains a list of cluster nodes inside the cluster leaders that keeps changing will consume too much energy. Moreover, the cluster leaders will be busy marinating the cluster structure most of the time. The limitation of mobility should be studied and new solutions that might involve less information maintained by the luster leaders need to be studied.

One different problem is that we try to do rekeying whenever the cluster leader suspects that a node was captured. This might be applicable in ad-hoc networks, but not in sensor networks. The reason is that nodes in senor networks just go to sleep and then wake up frequently to maintain their limited energy. Once the cluster leader do not hear from a node, it assumes it was captured and proceed with rekeying even if this node went temporarily to sleep. When the node wakes up, it assumes it is a new node and may rekey to maintain the backward secrecy. This unnecessarily rekeying consumes most of the energy in a network with limited energy such as sensor network.

One partial solution to this problem might be to ask the nodes to send a special message to the cluster leader when they are going to sleep and another message when they wake up. The energy consumed in this solution might be large but compared the energy consumed in doing unnecessary rekeying might be less. Although this solution might consume less energy, it is not scalable with respect to the number of nodes in the cluster. Energy consumption and scalability needs to be studied in more details.

Currently, the base station distributes keys to nodes in initialization. Capturing a node is handled through other nodes with no cluster leaders involved. Capturing the cluster leader or the base station might give the attacker enough information to threat the network. Two possible solutions for this problem: The first solution is to have tamper-resistant nodes that erase all the information they carry once they are captured. The other solution is the ability of handling node capturing in a fully distributed fashion. Without the involvement of cluster leader or base-station

Next generation sensor networks will be long-lived, highly dynamic with roaming nodes and multiple network replenishments to satisfy evolving QoS requirements. The attack profile on these networks will be more varied and complex. Research is needed on adaptive key management to address these new challenges. Also, it is to be noted that currently the presence of an intrusion detection system is assumed by both static and dynamic key management schemes. Intrusion detection in sensor networks remains an open research area at the time of this writing.

REFERENCES

[1] I.F. I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks", IEEE Comm. Magazine, August, 102-114(2002).

[2] L. Al-Sulaiman. "Cashing Techniques for increasing the availability of Mobile Ad-hoc Networks Certificate Authority services", Ph.D. Thesis, Old Dominion University, 2004.

[3] W. Su Akyildiz, , Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, 38(4), pp. 393—422, 2002.

[4] R. Anderson, H. Chan, and A. Perrig, "Key Infection: Smart Trust for Smart Dust," *12th* IEEE Int. Conf. Network Protocols (ICNP'04), 2004, pp. 206-212.

[5] Koichiro Ban and others, "Wireless Ad-hoc Networks Project", National Institute of Standardization and Technology (NIST), http://www.antd.nist.gov/wctg/manet/.

[6] J. Binder, J. King, K. Mooney, and M. Wilkinson, "A Lightweight Security Paradigm for Ad-hoc Wireless Networks", January 30, 2003.

[7] G. R. Blakley. "Safeguarding Cryptographic Keys". Proceedings of AFIPS 1979 National Computer Conference, Vol. 48, pp. 313-317, 1979.

[8] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences", In Advances in Cryptology – CRYPTO '92, LNCS 740, pp 471–486, 1993.

[9] B. Bhargav, X. Wu, Y. Lu, and W. Wang. "Integrating Heterogeneous Wireless Technologies: A Cellular Aided Mobile Ad-hoc Network (CAMA)," MONET Special Issue on Integration Heterogeneous Wireless Technologies 2004.

[10] E. Bommaiah, M. Liu, A. MvAuley, and R. Talpade. "AMRoute: Ad-hoc Multicast Routing Protocol", Internet Draft, draft-manet-amroute-00.txt.

[11] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes. "PGP in Constrained Wireless Devices" In 9th USENIX Security Symposium, pp 247261, August 2000.

[12] R. Canetti, T. Malkin, and K. Nissim. "Efficient communication-storage tradeoffs for multicast encryption." in Advances in Cryptology- EUROCRYPT '99, J. Stem, Ed. Lectures Notes in Computer Science, vol. 1599. Springer-Verlag, New York, pp. 459-474, 1999.

[13] D. Carman, P. Kruus, and B. Matt. "Constraints and Approaches for Distributed Sensor Network Security." NAI Labs Technical Report, 2000.

[14] M. Chorzempa, J. M. Park, and M. Eltoweissy, "SECK: Survivable and Efficient Keying in Wireless Sensor Networks," IEEE Workshop on Information Assurance in Wireless Sensor Networks, WSNIA'2005, April 2005.

[15] C. Cordeiro, H. Gossain, and D. Agrawal, "Multicast over Wireless Mobile Ad-hoc Networks: Present and Future Directions", IEEE Network, January/February 2003, Vol. 17, No. 1, pp. 52-59.

[16] H. Chan, A. Perrig, and D. Song, "Random Key Pre-distribution Schemes for Sensor Networks," Proc. IEEE Security and Privacy Symposium (ISPS'2003), 2003, pp. 197-213.

[17] H. Chan, A. Perrig, and D. Song. "Random key predistribution schemes for sensor networks." In IEEE Symposium on Research in Security and Privacy, 2003.

[18] Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. "Key management for secure internet multicast using boolean function minimization techniques," in Proceedings of Infocom'99, (New York), IEEE, March 1999.

[19] Y. Chawathe, S. McCanne, and E. A. Brewer. "RMX: Reliable multicast for heterogeneous networks." In Proceedings of IEEE Infocom, pp 795–804, 2000.

[20] Y. Chu, S. G. Rao, and H. Zhang. "A Case for End System Multicast." In ACM Sigmetrics 2000, Santa Clara, California, USA, 2000.

[21] C. M. Cordeiro and D. P. Agrawal, "Mobile Ad-hoc Networking", Tutorial/Short Course in 20 th Brazilian Symposium on Computer Networks, pp. 125-186, May 2002.

[22] S. Corson, J. Macker, RFC 2501 – "Mobile Ad-hoc Networking (MANET) Routing Protocol Performance Issues and Evaluation considerations", Network Working Group, RFC 2501.

[23] H. Deng, W. Li, and Dharma P. Agrawal. "Routing Security in Ad-hoc Networks." *IEEE Communications Magazine, Special Topics on Security in Telecommunication Networks*, Vol. 40 No. 10, October 2002.

[24] L. Dondeti, S. Mukherjee, and A. Samal. "Scalable secure one-to-many group communication using dual encryption." Computer Communications Vol. 23, No. 17, Nov. 1999, 1681-1701.

[25] W. Du, J. Deng, Y. S. Han, P. K. Varshney, "A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks," Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03), Washington D.C., October 2003.

[26] H. Deshpande, M. Bawa, and H. Garcia-Molina. "Streaming live media over a peer-to-peer network." Technical Report 2001-30, Stanford University (Computer Science Dept.), June 2001.

[27] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. "Deployment issues for the IP multicast service and architecture", IEEE Network Magazine, 2000.

[28] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," *Proc. of IEEE INFOCOM'04, March 2004.*

[29] B. Dutertre, S. Cheung, J. Levy, "Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust", *SDL Technical Report SRI-SDL-04-02*, April 6, 2004.

[30] M. Eltoweissy, H. Heydari, L. Morales, H. Sadborough., "Combinatorial Optimization of Key Management in Group Communications," *J. Network and Systems Management: Special Issue on Network Security, March 2004, p. 332b.*

[31] M. Eltoweissy, M. Younis and K. Ghumman, "Lightweight Multi-Granularity Key Management for Secure Wireless Sensor Networks," in the *Proceedings of the IEEE Workshop on Multi-hop Wireless Networks (MWN'04),* Phoenix, Arizona, April 2004.

[32] M. Eltoweissy, A. Wadaa, S. Olariu, and L. Wilson, "Group Key Management Scheme for Large-Scale Wireless Sensor Network," *J. Ad-Hoc Networks, September 2005, pp. 796-802.*

[33] M. Eltoweissy, M. Moharram, and R. Mukkamala, "Dynamic Key Management in Sensor Networks," IEEE Communications, April 2006.

[34] D. Eastlake, P. Jones, "US Secure Hash Algorithm 1 (SHA-1)," RFC 3174, IETF, September 2001.

[35] L. Eschenauer and V. Gligor, "A Key Management Scheme for Distributed Sensor Networks," in the Proceedings of the 9th ACM Conference on Computing and Communication Security, November 2002.

[36] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in the Proceedings of the 5th IEEE/ACM Annual Conference on Mobile Computing and Networks (MobiCOM'99), Seattle, WA, August 1999.

[37] F. Farhmand, S. Navathe, G. Sharp, and P. Enslow, "Managing Vulnerabilities of Information Systems to Security Incidents.", College of Computing Georgia Institute of technology.

[38] P. Francis, Yoid: "Extending the Internet multicast architecture", April 2000. http://www.aciri.org/yoid/docs/index.html.

[39] P. Gemmell, "An introduction to threshold cryptography", CryptoBytes Technical Newsletter, Volume 2, No. 3 - Winter 1997, RSA Laboratories.

[40] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. "The Round Complexity of Verifiable Secret Sharing and Secure Multicast", ACM Symposium on Theory of Computing 2000.

[41] S. Ghanem, Logical Key Hierarchy Maintenance and Rekey Protocols, Ph.D. Thesis, Computer Science Department, Old Dominion University.2004.

[42] R. Gennaro, M. O. Rabin, and T. Rabin. "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography." In Proc. of 17th PODC, 1998.

[43] S. Ghanem and H. Abdel-Wahab, "A Secure Group Key Management Framework: Design and Rekey Issues," Proc. IEEE Symposium on Computers and Communications (ISCC'2003), 2003.

[44] C. Gui, and P. Mohapatra. "Efficient Overlay Multicast for Mobile Ad-hoc Networks," IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, LA, March 2003.

[45] G. Gaubatz, J. Kaps, and B. Sunar, "Public Key Cryptography in Sensor Networks" - Revisited. ESAS 2004: 2-18.

[46] G. Gupta and M. Younis, "Load-Balanced Clustering in Wireless Sensor Networks," in the Proceedings of the International Conference on Communication (ICC 2003), Anchorage, Alaska, May 2003.

[47] G. Gupta and M. Younis, "Fault-Tolerant Clustering of Wireless Sensor Networks," in the Proceedings of the IEEE Wireless Communication and Networks Conference (WCNC 2003), New Orleans, Louisiana, March 2003.

[48] Z. Haas, *et al.,* "Wireless Ad-hoc Networks, Encyclopedia of Telecommunications," J. Proakis, editor, John Wiley, 2002

[49] H. Harney and C. Muckenhirn. "Group Key Management Protocol (GKMP) Specification." RFC 2093, 1997.

[50] Z. J. Haas, *et a.l.*, "Wireless Ad-hoc Networks,Encyclopedia of Telecommunications", John Proakis, editor, John Wiley, 2002.

[51] D. Harris and D. Pepelko, "Security in Wireless Ad-hoc Networks," JMU technical report, 2003.

[52] M. Horton, et al., "Mica: The commercialization of microsensor motes," Sensors Online Magazine, April 2002.

[53] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. O'Toole. "Overcast Reliable multicasting with an overlay network". In Proceedings of the Fourth Symposium on Operating Systems Design and Implementation, pp 197–212, San Diego, CA, October 2000. USENIX Association.

[54] S. Jajodia, R. Mukkamala, M. Eltoweissy, "Secure multicast for mobile commerce applications: Issues and challenges", in Advances in Security and Payment Methods for Mobile Commerce, Wen-Chen Chu (Editor), Springer-Verlag, 2004.

[55] IETF Multicast Security (MSEC) Working Group http://www.securemuticast.org

[56] G. Jolly, M. Kuscu, P. Kokate, and M. Younis, "A Low-Energy Key Management Protocol for Wireless Sensor Networks," *Proc. the IEEE Symposium on Computers and Communications (ISCC'2003), June 2003, p. 335.*

[57] D. Johnson, D. Maltz, and Y. Hu. "The Dynamic Source Routing Protocol for Mobile Ad-hoc Networks (DSR)", IETF MANET Working Group Internet-Draft, April 2003, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt

[58] K. Jones, A.Wadaa, S. Olariu, L. Wilson, and M. Eltoweissy, "Towards a New Paradigm for Securing Wireless Sensor Networks," NSPW 2003, August 18th - 21st, 2003. Ascona, Switzerland, Proceedings New Security Paradigms Workshop 2003, pp 115-122.

[59] G. Judge and F. Takawira. "Spread-sprectrum CDMA packet radio MAC protocol using channel overload detection and blocking", ACM Wireless Networks Volume 6 , Issue 6,December 2000, pp.: 467 - 479

[60] R. H. Katz, J. M. Kahn and K. S. J. Pister, "Mobile Networking for Smart Dust," in the Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, WA, August 1999.

[61] J. Kohl and B. Neuman, "The Kerberos Network Authentication Service (V5)". RFC 1510, September 1993.

[62] K. Langendoen and N. Reijers, "Distributed Localization in Wireless Sensor Networks: A Quantitative Comparison," Computer Networks (Elsevier), special issue on Wireless Sensor Networks, August 2003.

[63] Y. W. Law, "Key Management and Link-layer Security of Wireless Sensor Networks: Energy-Efficient Attack and Defense," *PhD thesis, CTIT Ph.D.-thesis Series 05-75, Univ. of Twente, December 2005.*

[64] S. Lee, M. Gerla, and C. Chiang. "On-Demand Multicast Routing Protocol", IEEE WCNC'99, New Orleans, LA, September 1999, pp.1298-1304.

[65] X. Li, R. Yang, M. Gouda, and S. Lam, "Batch rekeying for secure group communications", Proc. Tenth International World Wide Web Conf., pp.525-534, 2001.

[66] J. Liebeherr and T. K. Beam, "HyperCast: A Protocol for Maintaining Multicast Group Members" in a Logical Hypercube Topology. Proc. First International Workshop on Networked Group Communication (NGC '99), in: Lecture Notes in Computer Science, Vol. 1736, pp. 72-89, 1999.

[67] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicasting with delaunay triangulation overlays", IEEE J. Select. Areas Commun., vol. 20, pp. 1472-1488, Oct. 2002.

[68] C. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks", in the Proceedings of the IEEE Journal on Selected Areas of Communications, Vol. 15, No. 7, 1997.

[69] G. Lin and G. Noubir. "Multicast over Multihop Wireless Ad-hoc Networks, MADNET: Workshop on Mobile Ad-hoc Networking and Computing", Sophia-Antipolis, France March, 2003.

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

[71] D. Liu and P. Ning, "Improving Key Pre-Distribution with Deployment Knowledge in Static Sensor Networks," *ACM Trans. on Sensor Networks (TOSN), 2005. pp 204 - 239.*

[72] D. Liu, P. Ning, and Wenliang Du, "Group-Based Key Pre-Distribution in Wireless Sensor Networks," *Proc. of 2005 ACM Workshop on Wireless Security (WiSe 2005), Sept. 2005, pp. 11-20.*

[73] G. Lin and G. Noubir. "Secure Multicast over Multihop Wireless Ad-hoc Networks", MADNET:Workshop on Mobile Ad-hoc Networking and Computing, Sophia-Antipolis, France March, 2003.

[74] P. McDaniel, A. Prakash, and P. Heneyman. "Antigone: A flexible framework for secure group communication.", In Proceedings of the 8th USENIX Security Symposium. (Washington, D.C. Aug. 1997.). 99-114.

[75] D. McGrew and A. Sherman. "Key establishment in large dynamic groups using oneway function trees." Tech. Rep. No. 0755 (May), TIS Labs at Network Associates, Inc., Glenwood, Md, 1998.

[76] R. Min, et al., "Low Power Wireless Sensor Networks," in the Proceedings of International Conference on VLSI Design, Bangalore, India, January 2001.

[77] C. Mitchell and F. Piper. "Key Storage in Secure Networks.", Discrete Applied Mathematics. Col. 21(1988). pp 215-228.

[78] S. Mittra, "Iolus: A framework for scalable secure multicasting." In Proceedings of the ACM SIGCOMM. Vol. 27, 4 (New York, Sept. 1997) ACM, New York, pp. 277-288.

[79] M. Moharrum, R. Mukkamala, and M. Eltoweissy. "Efficient Secure Multicast with Well-Populated Multicast Key Trees", The Tenth International Conference on Parallel and Distributed Systems, ICPADS 2004, Newport beach CA.

[80] M. Moharrum, R. Mukkamala, and M. Eltoweissy. "CKDS: An Efficient Combinatorial Key Distribution Scheme For Wireless Ad-Hoc Networks", International Performance Computing and Communications Conference, IPCCC 2004, Phoenix, Arizona.

[81] M. Moharrum, R. Mukkamala, and M. Eltoweissy. "TKGS: Verifiable Threshold-Based Key Generation Scheme in Open Wireless Ad-hoc Networks", the Thirteenth International Cconference on Computer, Communications and Networks (ICCCN 2004).

[82] M. Moharram, R. Mukkamala, and M. Eltoweissy, "Efficient Secure Multicast Communications with Well-balanced Dynamic Key Trees," IEEE International Conference on Parallel and Distributed Systems (ICPADS 2004), July 2004.

[83] M. Moharrum and M. Eltoweissy. "Key Management Schemes in Sensor Networks: dynamic versus static keying." ACM Workshop on performance evaluation of Wireless Ad-hoc, Sensor and Ubiquitous Networks (PE-WASUN 2005), Montreal, Candas, October 2005.

[84] C. de Morais Cordeiro, H. Gossain, and D. Agrawal, "Multicast over Wireless Mobile Ad-hoc Networks: Present and Future Directions", IEEE Network, January/February 2003, Vol. 17, No. 1, pp. 52-59.

[85] R. Mukkamala, M. Moharrum, and M. Eltoweissy, "A Novel Architecture for Secure Group Communication in Wireless Ad-Hoc Networks with Application-Level Multicast", in the Proceedings of Workshop on Trusted Internet, Bangalore, India, December 2004.

[86] R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers". Communications of the ACM Vol. 21, No.12, pp. 993-999, 1978.

[87] S. Olariu, A. Wadaa, L. Wilson, and M. Eltoweissy, "Wireless sensor networks - Leveraging the Virtual Infrastructure", IEEE Network, August 2004.

[88] D. Otway and O. Rees, "Efficient and Timely Mutual Authentication", Operating Systems Review, 21 (1987), 8-10.

[89] J. Pegon and M. Subbarao. "Simulation Framework for a Mobile Ad-hoc Network", *Proc. of OPNETWORK 1999*, Washington DC., Sept. 1999.

[90] C. Perkins and E. Royer, "Ad-hoc On-Demand Distance Vector Routing", Second IEEE Workshop on Mobile Computing Systems and Applications, pp.90-100, February 1999

[91] C. Perkins, "Ad-hoc On-Demand Distance Vector (AODV) Routing", Nokia Research Center E. Belding-Royer, University of California, http://www.ietf.org/rfc/rfc3561.txt.

[92] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. "SPINS: Security Protocols for Sensor Networks." In Seventh Annual ACM International Conference on Mobile Computing and Networks(Mobicom 2001), Rome Italy, July 2001.

[93] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. "Efficient and Secure Source Authentication for Multicast", Network and Distributed System Security Symposium, NDSS'01, Feb. 2001.

[94] A. Perrig, D. Song, and D. Tygar. "ELK, a new protocol for efficient large-group key distribution," in: Proceedings of the IEEE Security and Privacy Symposium 2001, May 2001

[95] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS: Security Protocols for Sensor Networks", Wireless Networks Journal (WINE), September 2002.

[96] A. Perrig and H. Chan, "PIKE: Peer Intermediaries for Key Establishment in Sensor Networks.", IEEE Infocom 2005.

[97] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," Communications of the ACM, 43(5), pp. 51--58, 2000.

[98] S. Rafaeli and D. Hutchison. "A survey of key management for secure group communication", ACM Computing Surveys, Volume 35, Issue 3 , Pages: 309 - 329 , 2003.

[99] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. "Application-level multicast using content-addressable networks", In Proceedings of Third International Workshop on Networked Group Communication (NGC '01), London, England, 2001.

[100] E. Royer and C. Perkins. "Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol", Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99), Seattle, WA, USA, August 1999, pages 207-218.

[101] J. Rabaey, M. Ammer, J. Silva Jr., D. Patel, and S. Roundy, "PicoRadio supports ad-hoc ultra low power wireless networking," IEEE Computer, 33(7), pp. 42—48, 2000.

[102] R. Rivest, "The MD5 Message-Digest Algorithm", RFC 1320, MIT and RSA Data Security, Inc., April 1992.

[103] S. Setia, S. Zhu, and S. Jajodia. "A Comparative Performance Analysis of Reliable Group Rekey Transport Protocols for Secure Multicast", In Performance Evaluation, special issue Proceedings of Performance 2002, Rome, Italy, Sep., 2002.

[104] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie, "Protocols for self-organization of a wireless sensor network," IEEE Personal Comm., 7(5), pp. 16--27 2000.

[105] A. Shamir. "How to Share a Secret". Communications of the ACM, 22[11]:612- 613, November 1979.

[106] M. Srivastava. "Power Considerations for Sensor Networks", Collaborative Signal Processing Workshop January 14-16, 2001 Xerox PARC, Palo Alto, CA.

[107] J. Tavernier and M. Eltoweissy, "Effects of Keying Protocols on In-Network Processing in Wireless Sensor Networks," in the Proceedings of the IEEE Workshop on Algorithms for Wireless and Mobile Networks (A-SWAN 2004), Boston, August 2004.

[108] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. La Porta, "*Establishing Pair-wise Keys for Heterogeneous Sensor Networks," Technical Report, The Pennsylvania State University, 2004.*

[109] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless microsensor network models," ACM Mobile Computing and Communication Review (MC2R), 6(2), pp. 1—8, 2002.

[110] A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones, "Training a Sensor Network", Mobile Networks and Applications (MONET), June 2004

[111] A. Wadaa, S. Olariu, L. Wilson, and M. Eltoweissy. "WiSe: A Group Key Management Scheme for Wireless Sensor Networks", IEEE Mediterranean Electro-technical Conference, Croatia , May 2004.

[112] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. "The VersaKey framework: Versatile group key management". IEEE J. Sel. Areas Commun. (Special Issue on Middleware) Vol. 17, No. 9, Aug.1999, 1614–1631.

[113] D. Wallner, E. Harder, and R. Agee. "Key Management for Multicast: Issues and Architectures". RFC 2627, 1999.

[114] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz. "Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks", Third IEEE International Conference on Pervasive Computing and Communications (PERCOM'05).

[115] P. Wing and B. O'Higgins. "Using Public-Key Infrastructures for Security and Risk Management". IEEE Communications Magazine vol. 37 No.9 Sept. 1999, pp. 71-83.

[116] C. Wong, M. Gouda, and S. Lam. "Secure group communications using key graphs". IEEE/ACM Trans. Netw. 8, 1 (Feb. 2000), 16-30.

[117] J. Xie, R. Talpade, A. Mcauley, and M. Liu. "AMRoute: ad-hoc multicast routing protocol" , ACM Mobile Networks and Applications, Volume 7, Issue 6, December 2002

[118] Y. Yang, X. Li, X. Zhang, and S. Lam, "Reliable group rekeying: a performance analysis". SIGCOMM 2001: 27-38

[119] A. Yasinsac, V. Thakur, S. Carter, and I. Cubukcu. "A Family of Protocols for Group Key Generation in Ad-hoc Networks" Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN02), Nov 3-4, 2002

[120] R. Yavatkar, J. Friffioen, and M. Sudan. "A Reliable Dissemination Protocol for Interactive Collaborative Applications." In: ACM Multimedia 1995, pp. 333-343. November 1995.

[121] M. Younis, M. Youssef, and K. Arisha, "Energy-Aware management in Cluster-Based Sensor Networks," Computer Networks, Vol. 43, No. 5, pp. 649-668, December 2003.

[122] M. Younis, K. Ghumman, and M. Eltoweissy, "Efficient Location-aware Key management in Wireless Sensor Networks", Virginia Tech. Technical Report, March 2005.

[123] M. Younis, K. Ghumman, and M. Eltoweissy, "Location-aware Combinatorial Key Management Scheme for Clustered Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems, 2006.*

[124] M. Younis, K. Ghumman, and M. Eltoweissy, "Key Management in Wireless Ad-hoc Networks: Collusion Analysis and Prevention," IEEE (IPCCC'05), Phoenix, Arizona, April 2005.

[125] A. Youssef, A. Agrawala, and M. Younis, "Accurate Anchor-Free Localization in Wireless Sensor Networks," in the Proceedings of the 1st IEEE Workshop on Information Assurance in Wireless Sensor Networks (WSNIA 2005), Phoenix, Arizona, April 7-9, 2005.

[126] S. Yi and R. Kravets. "MOCA, a MObile Certification Authority for wireless networks", proceedings of the 2nd Annual PKI research workshop, NIST 2003.

[127] O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-hoc Sensor Networks," IEEE Trans. on Mobile Computing, Vol. 3, No.4, pp. 366- 379, Oct.-Dec. 2004.

[128] X. Zhang, S. Lam, D. Lee, and Y. Yang. "Protocol Design for Scalable and Reliable Group Rekeying" In IEEE/ACM Transactions on Networking, December 2003.

[129] W. Zhang and G. Cao, "Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration-Based Approach," IEEE INFOCOM, March 2005.

[130] L. Zhou and Z. Haas. Securing Ad-hoc Networks. IEEE Networks, Special Issue on Network Security. November/December, 1999

[131] L. Zhou, F. Schneider, and R. Renesse. "COCA: A Secure Distributed On-line Certification Authority". ACM Transactions on Computer Systems 20, 4 (November 2002), 329--368. Earlier version: Technical Report TR 2000-1828, December 7, 2000.

[132] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pair-wise keys for secure communication networks: a probabilistic approach," Proc. 11th IEEE International Conference on Network Protocols, Atlanta, Georgia, November 4-7, 2003

[133] S. Zhu and A. Chan. "Optimal Tree Structure for Key Management of Simultaneous Join/Leave in Secure Multicast," to appear in MILCOM 2004 Military Communications Conference, Monterey, CA.

[134] S. Zhu, S. Setia, and S. Jajodia. "LEAP: Efficient security mechanisms for large-scale distributed sensor networks", Proc. 10th ACM Conf. On Computer and Communications Security, Washington, DC, October 27-31, 2003, pages 62-72.

APPENDIX A

INITIALIZATION PROCEDURE AND INFORMATION KEPT BY EACH NODE IN DCK

Node	Pre-deployment	Initialization	Information Stored (after initialization)
Sensor Node (SN)	1-Unique Ids (optional GPS receiver) 2-Key generator functions KGF_b, KGF_c 3- Deployment Instance Seeds S_b, S_c	3- Generate k_b+m_b Keys using $KGF_b(S_b, K_0^b)$ 8- $K_{sc}^i(ID,loc) \rightarrow CL_i$ 11- Generate $k_{ci}+m_{ci}$ Keys using $KGF_c(S_c, K_i^c)$, *purge* m_{ci} keys 12-Purge the EBS_{ci} matrix except node's own key row 14- Purge K_i^*	k_{ci} bit EBS_{ci} Key string $K_0 \ldots K_{kci}$: k_{ci} *EBS keys* K_i^j: current shared key with BS
Key Generator Nodes (KGN)		3- Generate k_b+m_b Keys using $KGF_b(S_b, K_0^b)$ 8- $K_{sc}^i(ID,loc) \rightarrow CL_i$ 11- Generate $k_{ci}+m_{ci}$ Keys using $KGF_c(S_c, K_i^c)$, *purge* m_{ci} keys 12-Purge the EBS_{ci} matrix except node's own key string 14- Purge K_i^*	k_{ci} bit EBS_{ci} Key string $K_0 \ldots K_{kci}$: k_{ci} *EBS keys* K_i^j: current shared key with BS $C(k_{ci}+m_{ci}-1, m_{ci}-1)$* ID size
Cluster Leader (CL)		*1-*K_{sb}= $KGF_b(S_b,0)$ K_{sb}(ID, loc)→BS 3- Generate k_b+m_b Keys using $KGF_b(S_b, K_0^b)$ *4-* According to the key string, purge m_b keys 7- $K_i^*(K_{sc}^i, KS(CL_i))) \rightarrow$ All Sensor Nodes 9- *Determine* N_c^i, *and compute* k_{ci}, m_{ci}, EBS_{ci} Matrix, and selects the KGNs *10-* K_{sc}^i (k_b,m_b, K_c^i,*KGNs, key assignment*)→ All Sensor Nodes 11- Generate $k_{ci}+m_{ci}$ Keys using $KGF_c(S_c, K_i^c)$, *purge* m_{ci} keys 13- $K_{sc}^i(K_i^i) \rightarrow$*All Sensor nodes*	K_b bit EBS_b Key string k_{ci} bit EBS_{ci} Key string N_{ci} *Ids (and locations)* *A chain of t* future encrypted BS keys

Node	Pre-deployment	Initialization	Information Stored (after initialization)
Base Station (BS)		*2-Determine N_b, and compute k_b and m_b* *3- $K_{sb}(k_b, m_b, K_0^b)$* → All Nodes 5-For each cluster *i*, generate K_{sc}^i, and $\{K_i^j\}$ chain of *t* keys starting with $K_i^0 = K_i^*$, one of the keys purged by CL_i. 6- $K_0\ K_1 .. K_{bc}(\{K_i^j\}, K_{sc}^i, K_i^*(K_{sc}^i, KS(CL_i)))$ → CL_i, for each CL_i, with $K_0\ K_1 .. K_{bc}$ the EBS_b keys known to CL_i	EBS_b matrix and keys Current shared session key with each cluster *i*

APPENDIX B

MATHEMATICAL PROOFS OF USED FORMULAS

Lemma 0:

For integers m, n > 0, and arbitrary integers r,s, The following holds:

$$\sum_{i}\binom{n}{r+i}\binom{m}{s+i}=\binom{n+m}{n-r+s}$$

Proof:

The classic combinatorial correlations states that:

$$\sum_{i\geq 0}\binom{n}{r+i}\binom{m}{s-i}=\binom{n+m}{r+s}$$

A combinatorial justification for combinatorial correlation goes as follows: We have a population of n men and m women and want to select a committee of r+s people such that at least r members are men and at least s members are women. The seen in the left hand side and the excursion in the right hand side are two alternate ways of expressing the details of the committee that can be formed.

We can write:

$$\sum_{i=0}\binom{n}{r+i}\binom{m}{s+i}=\sum_{i}\binom{n}{r+i}\binom{m}{m-s-i}=\binom{n}{r+i}\binom{m}{(m-s)-i}=\binom{n}{r+i}\binom{m}{s'-i}$$

Where s'=m-s.

By combinatorial correlation, the last term in the above expression can be written as:

$$\sum_{i}\binom{n}{r+i}\binom{m}{s'-i}=\binom{m+n}{r+s'}=\binom{m+n}{r+m-s}=\binom{m+n}{r+m-(r-m+s)}=\binom{m+n}{n-r+s}$$

As claimed above in Lemma 0.

Corollary 0:

For all integers m, n ≥ 0,

$$\sum_i \binom{n}{i}\binom{m}{i} = \binom{n+m}{n}$$

Corollary 1:

If k=0; s=-1, Then:

$$\sum_i \binom{n}{i}\binom{m}{i-1} = \binom{n+m}{n-1}$$

Lemma 1:

For all integers m, k ≥ 0,

$$\sum_i^{\min\{k,m\}} \binom{n}{i}\binom{m}{i} i = k\binom{m+k-1}{k}$$

Proof:

We can write:

$$i.\binom{k}{i} = i\frac{k!}{i!(k-i)!} = \frac{k(k-i)!}{(i-1)!((k-1)-(i-1))!} = k\binom{k-1}{i-1}$$

Using the above formula

$$\sum_i^{\min\{k,m\}} \binom{n}{i}\binom{m}{i} i = k\sum_{i=1}^{\min\{k,m\}} \binom{m}{i}\binom{k-1}{i-1} = k\sum_i \binom{m}{i}\binom{k-1}{i-1}$$

By coronary 1, the right hand side can be written as

$$k\binom{m+k-1}{m-1} = k\binom{m+k-1}{k}$$

As claimed above in Lemma 1.

Theorem 1:

$$\sum_{i=0}^{\min\{k,m\}} \binom{m}{i}\binom{k}{i}(k-i+1) = \left[1+\frac{k^2}{m+k}\right]\binom{m+k}{k}$$

Proof:

The summation can be written as:

$$\sum_{i=0}^{\min\{k,m\}} \binom{m}{i}\binom{k}{i}(k-i+1) = \sum_i \binom{m}{i}\binom{k}{i}(k-i+1) = (k+1)\left(\sum_i \binom{m}{i}\binom{k}{i} - \sum_i \binom{m}{i}\binom{k}{i} i\right)$$

By applying Corollary 0 and Lemma 1, this expression becomes:

$$\sum_{i=0}^{\min\{k,m\}} \binom{m}{i}\binom{k}{i}(k-i+1) = (k+1)\binom{m+k}{k} - k\binom{m+k-1}{k}$$

Noticing that $\binom{m+k-1}{k} = \frac{m}{m+k}\binom{m+k}{k}$, then the above expression becomes

$$(k+1)\binom{m+k}{k} - \frac{km}{k+m}\binom{m+k}{k} = \left[k+1-\frac{km}{m+k}\right]\binom{m+k}{k} = \left[1+\frac{k^2}{m+k}\right]\binom{m+k}{k}$$

As claimed above in Theorem 1.

VITA

MOHAMMED A. MOHARRUM was born in Alexandria, Egypt, on December 30, 1974. He received his Bachelor of Science in Computer Science and Automatic Control from the Faculty of Engineering, Alexandria University, Egypt, in June 1997. He worked as a Research Assistant for the Informatics Research Institute at Muabarak City for Scientific Research from December 1997 to July 2000. In May 2000, he received his Master of Science from the Department of Computer Science and Automatic Control, Faculty of Engineering, Alexandria University, Egypt. He started working on his Ph.D. degree in Computer Science at Old Dominion University, in August 2000. During the course of his Ph.D. study, he co-authored twelve scientific conference and journal papers and a book chapter. He is working as a faculty member in the computer science department at Old Dominion University since 2005.

Permanent address: Department of Computer Science
Old Dominion University
Norfolk, VA 23529-0162
USA