2009

A Systematic Security Approach in Wireless Mesh Networks

Xia Wang
Iowa State University

A systematic security approach in wireless mesh networks

by

Xia Wang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Johnny S. Wong, Major Professor
Ying Cai
Wensheng Zhang
Yong Guan
Thomas E. Daniels

Iowa State University

Ames, Iowa

2009

DEDICATION

To my parents and my husband -
Without whose love and support I would not have been able to complete this work

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I am extremely lucky that I have support, encouragement, and inspiration from many people, without them this work would not have been possible.

My greatest gratitude goes to my advisor Dr. Johnny S. Wong for his guidance and consistent support. His knowledgeable, wise and inspiring discussions has guided me through my whole Ph.D. career. It was such a pleasure to work with him for all these years. Facing so many obstacles, I am lucky that he has always been there to show me the right direction and influenced me as an active thinker. Thank you, Prof. Wong!

I would also like to thank other members of my committee, Dr. Yong Guan, Dr. Thomas E. Daniels, Dr. Ying Cai, and Dr. Wensheng Zhang for their time and input. I would additionally like to thank Dr. Guan, Dr. Daniels and Dr. Ying Cai for their guidance on my initial stages of research and Dr. Wensheng Zhang for his insightful comments on my key management project. I would specially thank Dr. Samik Basu for his enlightening guidance and discussion on my research.

I am greatly thankful to my labmates, Chris Strasburg, Fred Stanley, Tanmoy Sarkar, and Ryan Michael Babbitt for their collaboration during this research and contributions to this dissertation. I wish to thank Tsing-yi Jiang, Yaping Jing, Taiming Feng, Tian Jiang, Wei Zhang, Ge Xu, Chuang Wang for their support and assistance during my study.

Finally, my deep gratitude goes to my husband (Dr. Tan Guo) and my parents (Fagui Wang and Chuxiu Liu) for their love, sacrifice and support during my life.

ABSTRACT

Wireless mesh networking has emerged as a key technology to provide wide-coverage broadband networking. It benefits both service providers with low cost in network deployment, and end users with ubiquitous access to the Internet from anywhere at anytime. Wireless mesh networks are vulnerable to malicious attacks due to the nature of wireless communication and the lack of centralized network infrastructure. Meanwhile, the capacity of multi-radio multi-channel communication, the need for heterogeneous network integration, and the demand for multi-hop wireless communication often make traditional security mechanisms inefficient or infeasible. Therefore, wireless mesh networks pose new challenges and call for more effective and applicable solutions.

In this work, we identify the requirement for a systematic security framework to protect wireless mesh networks and provide a security system with heterogeneity-aware intrusion prevention mechanism, cross-layer based intrusion detection technique, and a generic intrusion response model.

Our major contributions lie in the following: (1) We identify the architecture heterogeneity of wireless mesh networks and proposed a novel heterogeneity-aware group key management framework which combines the logical key hierarchical technique together with the localized threshold-based technique. (2) To leverage link-aware routing characteristics, we present a cross-layer based anomaly detection model which utilizes machine learning algorithms for profile training and intrusion detection. (3) We address the automatic intrusion response problem in wireless mesh network by providing a generic response model to describe the dependency of system services and resources. The dependency graph is later used for damage cost

assessment and response cost evaluation. (4) We build a wireless mesh network testbed and implemented a system prototype for intrusion detection system. Our simulation and experiment results show that our solutions outperform existing ones and are practical for wireless mesh networks in terms of communication overhead and performance speed.

CHAPTER 1. OVERVIEW

1.1 Introduction

Wireless mesh networking has emerged as a key technology to provide wide-coverage broadband networking. It benefits both service providers with low cost in network deployment, and end users with ubiquitous access to the Internet from anywhere at anytime [15]. A wireless mesh network (WMN) is composed of a wireless infrastructure and associated client networks. The wireless infrastructure, normally called wireless mesh backbone, contains a set of mesh nodes (or mesh routers) which could be either static or dynamic and self-form a multi-hop wireless ad hoc network to relay data from client networks. A wireless device can either directly connect to a mesh router or indirectly associate with the backbone through various wireless networks, such as wireless ad hoc networks, sensor networks, and WiFi networks. Such integration is accomplished through the routing and gateway functionalities provided by mesh routers.

Due to its promising future, WMNs have gained a lot of attention from both academia and industry. Many testbeds are built for research purposes and commercial products are available for real WMNs. However, many issues have to be solved before its widely deployment. For instance, the available MAC and routing protocols are not scalable; throughput drops significantly as the number of nodes or hops in WMNs increases. Especially, communication security and privacy over WMNs are big concerns due to its vulnerability to various malicious attacks. For instance, adversaries can eavesdrop on wireless communication to gain confidential information. Through compromised nodes, attackers can launch DoS attacks and modify

the content of transmitted information, thus jeopardize the confidentiality, authenticity, availability and integrity of the whole network.

Like mobile ad-hoc networks (MANETs), WMNs have the properties of shared medium, lack of traffic aggregation point, and dynamic topology. Due to those characteristics traditional designs of security mechanisms in wired networks can not be directly applied to WMNs. In addition, the mechanisms used in MANETs are not suitable for WMNs, either. This is due to the reason that WMNs diversify MANETs in many aspects. Mesh nodes are usually equipped with multiple radios. Thus, multiple channels are assigned at each node to support simultaneous data transmission and reception [16, 60]. The integration of various types of wireless networks also requires heterogeneity awareness design in intrusion detection. Further, new challenges are presented in network protocols which integrates link information in routing selection to improve the performance of multi-hop wireless transmission [37]. All those present new challenges for security mechanism in WMNs.

In this work, we provide a systematic security framework which includes intrusion prevention, intrusion detection, and intrusion response system to secure WMNs.

Intrusion prevention embeds security design with the specific mechanism such that possible intrusions may be avoided. In this system, we study group key management in WMNs. We identify the architecture heterogeneity of WMNs and proposed a novel heterogeneity-aware group key management framework which combines the logical key hierarchical technique together with the localized threshold-based technique. Specifically, a WMN multicast session involves both static backbone nodes and mobile client nodes: for backbone nodes which are topologically-stable but may spread over a large area, the localized threshold-based group key management technique is applied since the technique requires stable network topology and its rekeying delay is independent of network scale; for client nodes associated with each backbone node which are mobile but confined in a limited area, the logical key hierarchical technique is applied since the technique is resilient to node mobility but its rekeying delay is proportional to the network scale.

Intrusion prevention is the first protection line in a security system. However, once the first defense line is penetrated, intrusion detection system is required to detect attacks and generate alerts. Network intrusion detection often concentrated on analyzing network traffic. In WMNs, new routing metrics are designed to accommodate multi-radio and multi-channel characteristics and provide high performance throughput. The wireless link quality is integrated with routing selection. In this case, cross-layer design is a necessity of intrusion detection design. We present a cross-layer based anomaly detection model which utilizing machine learning algorithms for profile training and intrusion detection. Statistical features from both network layer and data link layer are collected and processed. We have implemented the anomaly detection model on a WMN testbed. Experimental results have shown that cross-layer based method has higher detection rate and lower false alarm rate on the average, comparing to single-layer (e.g. network layer) intrusion detection.

Intrusion detection is often followed by intrusion response actions which are dedicated to minimize the effects of intrusion. Existing responses are limited to manual response which requires network administrator to manually select actions in detection of intrusions. Among automatic intrusion responses, a generic response model is required in describing the dependency of system services, and resources, and determine the damage of intrusion and cost of response, therefore, a proper response action can be selected.

1.2 Wireless Mesh Network

1.2.1 Network Architecture

The network architecture of a typical WMN is presented in Fig. 1.1. Generally, a WMN includes two components: wireless infrastructure/backbone and mesh clients. The wireless infrastructure/backbone is a mesh with self-configuring, self-healing links among a set of mesh routers. It can be built using various radio technologies, in addition to the mostly used IEEE 802.11 technologies. The wireless infrastructure/backbone is connected to the Internet

through one or more mesh routers with gateway functionalities. Mesh clients with the same radio technologies as mesh routers can directly connect to the backbone. Others can associated with mesh routers through client networks such as cellular networks, sensor networks, wireless ad hoc networks, and WiMAX [77]. The integration of different client networks within WMNs are accomplished by employing the gateway and bridging functionalities in mesh routers.

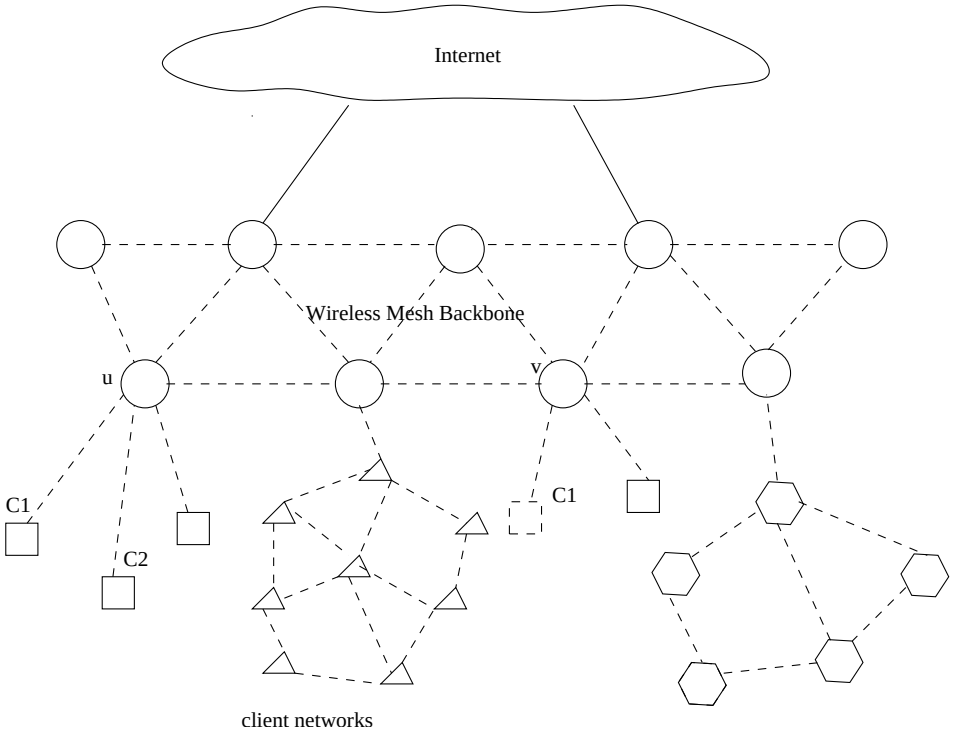


Figure 1.1 Network architecture of a wireless mesh network

1.2.2 Characteristics

Like wireless ad-hoc networks, WMNs use radio signal as its communication media. They generally have distributed infrastructure and lack centralized traffic aggregation points. Network topology may dynamically change due to addition or deletion of mesh nodes. Hence, WMNs are generally considered to be a type of ad-hoc networks. However, WMNs provide more capabilities than wireless ad-hoc networks. For instance, WMNs provide wide-area wireless network coverage.

In general, the characteristics of WMNs are summarized as follows:

Multi-hop wireless networks. One objective to develop WMNs is to extend the coverage range of current wireless networks without sacrificing the channel capacity. Another objective is to provide non-line-of-sight (NLOS) connectivity among the users without direct line-of-sight (LOS) links. The mesh-style multi-hopping is indispensable.

WMNs support ad-hoc networking and capability of self-forming, self-healing, and self-organization. WMNs enhance network performance due to its flexible architecture, easy deployment and configuration. Its mesh connectivity provides multipoint-to-multipoint communication and fault tolerance to the network. Due to those features, WMNs have low up-front investment requirement and the network can grow gradually as needed.

Heterogeneity. Mesh routers integrate heterogeneous networks. Thus multiple types of network access exist in WMNS. Wireless mesh backbone in WMN provides the infrastructure for interconnect different types of networks. Both backhaul access to the Internet and peer-to-peer (P2P) communication are supported.

Compatibility and interoperability with other wireless networks. For example, WMNs built on IEEE 802.11 technologies must be compatible with IEEE 802.11 standards in the sense of supporting both mesh capable and conventional Wi-Fi clients. WMNs also need to be inter-operable with other wireless networks such as WiMAX and cellular networks.

Mobility and power consumption constraints are different for mesh routers and mesh clients. Mesh routers are usually stationary and do not have strict power constraints, whereas mesh clients are mobile and they can roam between different WMN domains and are usually battery powered.

1.2.3 Applications

The research and development of wireless mesh networking has driven by key applications that can be widely deployed. Among them several applications have clearly demonstrated

their promising market value. For example, broadband home networking, community and neighborhood networking, enterprise networking, metropolitan area networking, transportation system, and health and medical systems.

The current broadband home networking relies on a single access point which is wired to the Internet. Often there are dead zones in a house and it is costly to add more access points to increase coverage. In addition, communication through different access points are not efficient. WMNs can solve all these problems. By replacing the access point with multiple mesh routers, a wide coverage and robust wireless home network can be built in which data communication are routed through the wireless mesh backbone.

A WMN can also be built for a community or between neighborhood for data sharing and improving connectivity. Similarly, WMNs can be applied for enterprise networking to provide wireless connection among offices or building. In large areas, such as cities, WMNs are economical alternative to broadband networking, especially in underdeveloped regions. Another application for WMN is to provide instant traffic information for passengers through infrastructure installed on trains, buses, and ferries. In addition, medical and health system will also benefit from the high bandwidth and easy access of WMNs.

1.2.4 Testbeds and Implementations

Thrilled by its value in fundamentally resolving the limitation and significantly improving the performance of wireless LANs, many universities and research institutes have built WMN testbeds to study practical issues in deploying such networks as well as to test and refine theoretical ideas to improve their practical applicability.

One of the earliest WMN testbed is Carnegie-Mellon University's *mobile ad hoc network testbed* [52]. It has seven nodes: two stationary nodes and five car mounted nodes that drive around the testbed site. One car simulates node entering and leaving the testbed with mounted roving node. The testbed adopts DSR routing protocol [44] for packet routing. The major

purpose of the testbed is to examine network behavior under different levels of traffic load, including audio and video streams, and to study protocol design enhancements.

MIT's *roofnet* [13, 14] is an experimental 802.11 b/g mesh network developed at MIT that provides broadband Internet access to users in Cambridge, MA. There are currently around 20 active nodes on the network. The testbed is used to examine link-level measurements of 802.11, to find high-throughput routes in the face of lossy links, to select adaptive bit rate, and to develop new protocols which take advantage of radio unique properties.

Berlin Roof Net (BRN) [1] is a project run by volunteer students of the Computer Science Department at Humboldt University, Berlin Germany. Mesh nodes are run independently by the students with their own equipment. The objective is to study protocol design in the self-organized and self-configuring mesh network in a large city as Berlin.

University of Illinois at Urbana-Champaign has built a WMN testbed to study various form of diversity available for multi-radio multi-channel WMNs. With multiple interfaces at each node, the network capacity is improved. However, with mesh nodes configured to the 3 non-overlapping channels in 802.11b standard, the performance does not reach 3 times of the single-channel network performance due to the inter-channel interference. Methods to eliminate radio interference have been studied.

The University of California, Santa Barbara MeshNet [12] is a mesh testbed composed of 25 nodes distributed on five floors in one building. Each node is equipped with multiple IEEE 802.11a/b/g wireless radios. This network is used to study scalable routing protocols, efficient network management, and multimedia streaming over multi-hop wireless networks.

The Broadband and Wireless Network (BWN) Lab at Georgia Institute of Technology has built a WMN testbed, called *MeshGIT* [2], which is composed of 15 IEEE 802.11b/g based mesh routers. The WMN testbed is connected to the next generation Internet testbed through a few mesh routers. On this testbed, experiments investigating the relationship between network performance and network parameters such as distance, clustering, and backhaul placement are conducted. Thus new protocols at different network layers and cross layer designs are

developed and evaluated.

The WMN testbed, MAP (Mesh at Purdue) [8], at Purdue University has been utilized to study practical issues such as routing protocols, applications, and network deployment.

In this work, we have built a WMN testbed to study network security issues. The testbed is distributed on different floors in two buildings. Each building has 10 nodes. Each node is a PC equipped with two wireless interface cards (NIC) and configured as a mesh router. We have implemented and evaluated our intrusion detection and response systems over this testbed.

Since WMN has been introduced some commercial products have been released. For instance, Cisco provides Aironet 1500 series Mesh Access Point [3] which is a dual radio system supporting a 2.4 GHz access link and a 5GHz transit link. Nortel produces Wireless Access Point 7220 [6] for wireless mesh network. Tropos Networks [11] employs a cellula Wi-Fi network architecture. It's layer-3 network operation system called Tropos Sphere runs on standard 802.11 hardware and software and allow Wi-Fi cells to inter-operate and form a completely wireless network. Strix Systems [10] provides Access/One Network products and solutions for wireless mesh network deployment in different environments. Microsoft Research Lab (MSR) has implemented ad hoc routing and link quality measurement in a software model called mesh connectivity layer (MCL) [4]. MCL implements routes by using a modified version of DSR called LQSR, which allows wireless nodes interconnected to form a mesh network. In the network stack, MCL sits between layer-2 and layer-3 and appears to be another protocol running over the physical link.

1.3 Taxonomy of Wireless Attacks in Wireless Mesh Networks

Similar to MANETs, wireless mesh networks are subject to various malicious attacks. Some of the possible attacks are summarized as follows:

- *Eavesdropping*: Adversaries use electronic transmitting or recording devices to monitor wireless communications to gain critical information. It is generally the first step in

launching further attacks in wireless networks.

- *Traffic Analysis*: Adversaries analyze traffic flows to deduce information from the patterns of wireless communication without cracking the security system of the wireless communication system.
- *Radio Jamming*: Adversaries transmit a high-power signal to disrupt or interfere with legitimate wireless communication.
- *Replay Attack*: Adversaries may replay messages received from other nodes or received previously to disturb the functionalities of wireless networks.
- *Rushing Attack*: Adversaries always forward ROUTE REQUEST packets more quickly than legitimate nodes in order to increase the probability that routes with attackers will be discovered rather than other valid routers [41].
- *Wormhole Attack*: Adversaries build a wormhole tunnel between two end points which are usually multi-hops away. The message recorded at one end point is relayed to the other end and re-broadcasted into the network, which fools the wireless nodes far from each other to believe they are neighbors.
- *Blackhole Attack*: An adversary node advertises itself as having the shortest path to the destination node whose traffic it wants to intercept. By doing this, the malicious node can deprive the traffic from the source node.
- *Packet Dropping/Selective Forwarding*: A compromised node may drop all or some of the messages that should be forwarded.
- *Packet Flooding*: Adversaries may send a huge amount of useless information to the network through compromised nodes to disrupt wireless communication.
- *Modification/Pollution*: Adversaries can modify or corrupt the messages transmitted in wireless networks.

- *Impersonating/Sybil Attack*: A compromised node can illegitimately claim identities of multiple legal nodes or can impersonate another legal node.

We categorize these attacks and provide the a taxonomy of wireless attacks based on different criteria [82].

- Attacks can have different security goals in terms of the well-known security requirements: *Confidentiality*, *Integrity*, *Availability*. For instance, eavesdropping and traffic analysis have the target to compromise confidentiality. Relay attack, wormhole attack, and modification/pollution affect the network integrity. Radio jamming, packet flooding, packet dropping/selective forwarding are examples of DoS attacks, which jeopardize the availability of network services.

- Attacks can be *passive* or *active*. Passive attacks are characterized by interception of messages without modification. Identifying the parties involved between the communication or the traffic pattern can be significant by itself. They are very hard to be detected and prevention is a priority. For example, traffic analysis is a passive attack and can not be protected using encryption mechanisms solely. Active attacks either generate new (malicious) information and/or modify existing one. One example is modification/pollution attacks.

- Malicious attacks may have different targets in the network. Communication information, wireless node, or network service are all possible targets. Identifying attack targets are critical in addressing these attacks. For example, the target of sybil attacks is wireless nodes identities and that of eavesdropping attack is communication information.

- Attacks may occur at different network layers and can be addressed using proper mechanisms at that layer. For example, radio jamming is a physical layer attack that can be addressed using advanced modulation techniques such as DSSS or FHSS, while sybil attack occur at application layer and can be countered using authentication approaches.

- Attacks can be intermittent or persistent from the perspective of how long they can last. Persistent attacks last long, but intermittent ones happen and disappear very quickly. So in most cases, intermittent attacks do not demand us to design specific countermeasures.

We note that some complicated attacks may even consist of multiple attacks from different categories. For example, an attack may start with eavesdropping. Once enough information is collected for security cracking, Modification/Pollution attack can be launched. Some attacks can happen at different layers at the same time. For instance, packet dropping is a DoS attack that can happen at both MAC layer and network layer. Cross layer attacks are even harder to be detected correctly. In network routing, adversaries can first attract traffic using wormhole attack, then use packet dropping to disrupt the routing service.

1.4 Challenges and Issues in Wireless Mesh Networks

Wireless mesh networks are vulnerable to various malicious attacks and their characteristics have determined that efficient mechanisms have to be provided to support a secure communication system. In our work, we aim to provide a systematic security framework to address intrusion prevention, intrusion detection and intrusion response problems in WMNs.

Group communication is a key application over WMNs. Group key management, including group key generation, distribution, revocation, and update, plays a critical role in securing the communication among group members. In WMNs, wireless routers are relatively stationary while wireless clients are mobile which may result with frequent group membership changes. On group member leaving and joining the group, key has to be refreshed. Traditional methods that relying on a centralized group server for key updating are not efficient as each key updating has to traverse the whole network. In addition, the heterogeneous network architecture has not been addressed in group key management. These problems motivated us to study group key management for wireless mesh networks.

Wireless mesh networks are generally equipped with multiple radio technologies for better performance in terms of network throughput. However, careful design has to be planned for radio selection and channel negotiation. To leverage multi-radio and multi-channel communication cross-layer design has become a necessity for WMNs. For instance, in routing selection, a path with multiple channels may achieve better performance than a path with the shortest hop count. Because data may get transmitted at the same time on consecutive links if each link has been properly configured to different channels. Inspired by that, we study cross-layer based design in intrusion detection.

Intrusion detection is generally followed by intrusion response actions. However, research on intrusion response is still in its infancy. Most existing intrusion response rely on network administrator to manually select and deploy response actions. Few automatic intrusion response has been studied. However, no generic response model has been proposed for damage assessment and response cost measurements. To provide a complete secure system, and to study a generic model for intrusion response, we investigate the intrusion response issue in WMNs.

1.5 Contribution of the Thesis

In this work, a systematic framework is provided for secure communication over WMNs. The framework includes three components: a heterogeneity-aware group key management to protect communication over the network, a cross-layer design of anomaly intrusion detection system, and a response model that offers a generic method in damage assessment and cost evaluation in selecting response actions. Our contributions lie in the following:

- We propose a novel heterogeneity-aware group key management framework which combines the logical key hierarchical technique together with the localized threshold-based technique. Specifically, a WMN multicast session involves both static backbone nodes and mobile client nodes: for backbone nodes which are topologically-stable but may

spread over a large area, the localized threshold-based group key management technique is applied since the technique requires stable network topology and its rekeying delay is independent of network scale; for client nodes associated with each backbone node which are mobile but confined in a limited area, the logical key hierarchical technique is applied since the technique is resilient to node mobility but its rekeying delay is proportional to the network scale.

- We present a cross-layer based anomaly detection model which utilizing machine learning algorithms for profile training and intrusion detection. Statistical features from both network layer and data link layer are collected and processed. We have implemented the anomaly detection model on a WMN testbed. Experimental results have shown that cross-layer based method has higher detection rate and lower false alarm rate on the average, comparing to single-layer (e.g. network layer) intrusion detection.

- we propose a generic model which describes the dependencies between services and resources in a system. The dependency relationship will be presented in a dependency graph. In this graph cost values of services or resources are propagated down which are later used for damage cost estimation and response cost evaluation.

1.6 Organization of the Thesis

The rest of the work is organized as follows. Chapter 2 gives a literature overview of security research in wireless mesh networks. Chapter 3 describes the heterogeneity-aware group key management. A framework is first introduced followed by an implementation. Chapter 4 addresses intrusion detection in WMNs utilizing cross-layer based design. A system prototype is implemented over a WMN testbed. Chapter 5 presents a generic model for intrusion response. A dependency graph of a system will be generated and used for damage measurements and response selection. In chapter 6, we summarize the work and discuss future works.

CHAPTER 2. REVIEW OF LITERATURE

2.1 Group Key Management in Wireless Mesh Networks

2.1.1 Introduction

Group key management plays a key role in securing communication over WMNs. Many applications over WMNs are built based on group communication model: packets are required to be delivered from one or more authorized senders to a large number of authorized receivers. In order to limit the access to the data being transmitted to group members, authentication or access control management have to be enforced in the operations. Group key management are thus deployed.

In a secure group communication, a number of nodes share a secret encryption key(s), called group key(s). Using the group key, a group member can encrypt its message and broadcast it into the network. Only the nodes that are in the same group can decrypt the message. Efficiency is achieved for such communication because data packets need to be transmitted once and they traverse any link between two nodes only once, hence saving bandwidth. This contrasts with unicast-based group communication where the sender has to transmit n copies of the same packet.

Group key management involves key generation, distribution, and key updating. A group key is a cryptographic key such that given the cipher text there is no easy way to recover the original message other than by knowing the correct key. The major issue in key generation is who is responsible for key generation: a single server or a group of nodes?

Furthermore, a group may require that membership changes cause the group key to be refreshed. Changing the group key prevents a new member from decoding messages exchanged before it joined the group. If a new key is distributed to the group when a new member joins, the new member cannot decipher previous messages even if it has recorded earlier messages encrypted with the old key. Additionally, changing the group key prevents a leaving or expelled group member from accessing the group communication (if it keeps receiving the messages). If the key is changed as soon as a member leaves, that member will not be able to decipher group messages encrypted with the new key.

However, distributing the group key to valid members is a complex problem. Although rekeying a group before the join of a new member is trivial (send the new group key to the old group members encrypted with the old group key), rekeying the group after a member leaves is far more complicated. The old key cannot be used to distribute a new one, because the leaving member knows the old key. Therefore, a group key distributor must provide another scalable mechanism to rekey the group.

A naive solution for rekeying a group with n members has the key distribution centre (KDC) assigning a secret key to each member of the group. In order to distribute the group key, the KDC encrypts it with each member's secret key. This operation generates a message O(n) long which is then transmitted to the whole group via multicast. On receiving the message, a member can recover the group key from the appropriate segment of the message using its own secret key.

Obviously, the naive solution is not scalable, and it frequently requires the use of secure channels. Secure channels are not always easy to establish. For these reasons different group key management protocols have been developed, each with different properties and performance.

2.1.2 Classification of Group Key Management

Group key management schemes can be divided into three main classes:

- *Centralized group key management protocols*. A single entity 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 architectures*. 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 generation. All members can perform access control and generation of the key can be either contributory, meaning that all members contribute some information to generate the group key, or done by one of the members.

2.1.3 Centralized Group Key Management

Wong et al. [79] and Wallner et al. [74] propose the use of a Logical Key Hierarchy (LKH). In this approach, a KDC maintains a tree of keys. The nodes of the tree hold key encryption keys (KEKs). The leaves of the tree correspond to group members and each leaf holds a KEK associated with that one member. Each member receives and maintains a copy of the KEK associated with its leaf and the KEKs corresponding to each node in the path from its parent leaf to the root. The key held by the root of the tree is the group key. For a balanced tree, each member stores at most $\log(n) + 1$ keys, where $\log(n)$ is the height of the tree. A joining member is associated with a leaf and the leaf is included in the tree. All KEKs in the nodes from the new leaf's parent in the path to the root are compromised and should be changed (backward secrecy). A rekey message is generated containing each of the new KEKs

encrypted with its respective node's children KEK. The size of the message produced will be at most $2 * \log(n)$ keys long. Removing a member follows a similar process. When a member leaves (or is evicted from) the group, its parent node's KEK and all KEKs held by nodes in the path to the root are compromised and should be updated (forward secrecy). A rekey message is generated containing each of the new KEKs encrypted with its respective node's children KEK. The exception is the parent node of the leaving member's leaf. The KEK held by this node is encrypted only with the KEK held by the remaining member's leaf. As the key held by the leaving member was not used to encrypt any new KEK, and all its known KEKs were changed, it is no longer able to access the group messages.

An improvement in the hierarchical binary tree approach is a one-way function tree (OFT) and was proposed by McGrew and Sherman [21]. Their scheme reduces the size of the rekeying message from $2 * \log(n)$ to only $\log(n)$. Here a node's KEK is generated rather than just attributed. The KEKs held by a node's children are blinded using a one-way function and then mixed together using a mixing function. The result of this mixing function is the KEK held by the node. This is represented by the following formula:

$$k_i = f(g(k_{left(i)}), g(k_{right(i)}))$$

Where $left(i)$ and $right(i)$ denote respectively the left and right children of node i. The function g is one-way, and f is a mixing function.

Canetti et al. [29] proposed a slightly different approach that achieves the same communication overhead. Their scheme uses a pseudo-random-generator to generate the new KEKs rather than a one-way function and it is applied only on user removal. This scheme is known as the one-way function chain tree.

Waldvogel et al. [73] extended their own solution proposing to change the hierarchical tree for a flat table (FT) with the effect of decreasing the number of keys held by the KDC. The table has one entry for the Traffic Encryption Key (TEK) and $2w$ more entries for KEKs, where w is the number of bits in the member id. There are two keys available for each bit in

the member id, one associated with each possible value of the bit. A member knows only the key associated with the state of its bit. In total, each member holds $w + 1$ keys.

2.1.4 Decentralized Group Key Management

RFC1949 [23] proposes a scheme to use the trees built by the Core Based Tree (CBT) multicast routing protocol to deliver keys to a multicast group. Any router in the path of a joining member from its location to the primary core can authenticate the member since the router is authenticated with the primary core. This scheme requires some modifications to the IGMP6 and assumes that CBT is deployed. Furthermore, there is no solution for forward secrecy other than to recreate an entirely new group without the leaving members.

Mittra proposes Iolus [54], a framework with a hierarchy of agents that splits the large group into small subgroups. A Group Security Agent (GSA) manages each subgroup. The GSAs are also grouped in a top-level group that is managed by a Group Security Controller. Iolus uses independent keys for each subgroup and the absence of a general group key means membership changes in a subgroup are treated locally. It means that changes that affect a subgroup are not reflected in other subgroups. In addition, the absence of a central controller contributes to the fault-tolerance of the system. If a subgroup controller (namely GSA) fails, only its subgroup is affected.

Dondeti et al. [36] proposed a dual-encryption protocol (DEP). In their work, they suggest a hierarchical subgrouping of the members where a subgroup manager (SGM) controls each subgroup. There are three kinds of KEKs and one Data Encryption Key (DEK). KEK_{i_1} is shared between a SGM_i and its subgroup members. KEK_{i_2} is shared between the Group Controller (GC) and the members of subgroup i, excluding SGM_i . Finally,GC shares KEK_{i_3} with SGM_i . In order to distribute the DEK to the members, the GC generates and transmits a package containing the DEK encrypted with KEK_{i_2} and encrypted again with KEK_{i_3}.

Setia et al. [62] proposed Kronos. It is an approach driven by periodic rekeys rather than

membership changes, which means a new group key is generated after a certain period of time, disregarding whether any member has joined, left or been ejected from the group. Although Kronos can be used within a distributed framework, it works differently because the DKD does not directly generate the group key. Instead, each AKD independently generates the same group key and transmits it to its members at the end of the predetermined period.

2.1.5 Distributed Group Key Management

Group DiffieHellman key exchange [51] is an extension for the Diffie-Hellman (DH) key agreement protocol that supports group operations. The DH protocol is used for two parties to agree on a common key. In this protocol, instead of two entities, the group may have n members. The group agrees on a pair of primes (q and α) and starts calculating in a distributive fashion the intermediate values. The first member calculates the first value (α^{x_1}) and passes it to the next member. Each subsequent member receives the set of intermediary values and raises them using its own secret number generating a new set. A set generated by the ith member will have i intermediate values with $i-1$ exponents and a cardinal value containing all exponents.

Boyd [] proposed yet another protocol for conference key agreement (CKA) where all group members contribute to generating the group key. The group key is generated with a combining function: $K = f(N1, h(N_2), ..., h(N_n))$, where f is the combining function, h is a one-way function, n is the group size and N_i is the contribution from group member i. The protocol specifies that $n-1$ members broadcast their contributions (N_i) in the clear. The group leader, for example U_1, encrypts its contribution (N_1) with the public key of each $n-1$ group member and broadcasts it. All group members who had their public key used to encrypt N_1 can decrypt it and generate the group key.

Zhang et al [83] present a distributed key management to address the node compromising problem and to improve the key updating performance in sensor network. In this group key

management, the polynomial to generate future keys are pre-deployed into individual nodes and then appropriately processed before it is discarded. Future group keys are calculated through the local collaboration among neighbors. Such localized collaboration demonstrated a method to reduce cost in key updating and also achieve a reasonable level of security.

Network topology is exploited in Sun et al's Topology-Matching Key Management (TMKM) [66] to reduce the rekeying message in cellular network. A TMKM tree is built such that neighboring nodes in the key tree are also physical neighboring nodes. To control message broadcast in cellular network, base stations (BSs) and super hosts (SHs) actually hold some of the Key Encryption Keys (KEKs) in the key tree and rekeying message is only broadcast by a SH to its governed BSs if and only if the rekeying message is useful to one or several BSs and by a BS to its subscribed users if and only if the rekeying message is useful to the users. In the above method, only the last hop is considered to be wireless.

2.2 Intrusion Detection in Wireless Mesh Networks

2.2.1 Introduction

Intrusion detection can be viewed as a passive defense, similar to a burglar alarm in a building. An intrusion detection system (IDS) attempts to differentiate abnormal activities from normal ones, and identify truly malicious activities (attacks) from the abnormal but non-malicious activities. Unfortunately, normal activities have a wide range of scenarios, and attacks may appear similar to normal activities. For example, a ping is a common utility to discover if a host is operating and online, but a ping can also be used for attack reconnaissance to learn information about potential targets. Even if unusual activities can be distinguished from normal activities, an unusual activity may not be truly malicious in intent.

Based on different criteria, IDSs can be divided into different categories. Depending on the monitored events IDSs can be classified into two types: host-based or network-based IDS. Host-based IDS are installed on hosts and monitor their internal events, usually at the operating

system level. These internal events are the type recorded in the host's audit trails and system logs. In contrast, network-based IDS monitor packets in the network. This is usually done by setting the network interface on a host to promiscuous mode (so all network traffic is captured, regardless of packet addresses). Alternatively, there are also specialized protocol analyzers designed to capture and decode packets at full link speed.

Based on analysis method in IDSs, three different techniques can be used: misuse detection, anomaly detection, and specification based detection. Misuse detection is also called signature-based detection because the idea is to represent every attack by a signature (pattern or rule of behavior). If a matching signature is found, that attack is detected. An advantage of misuse detection is its accuracy. If a signature matches with an attack, that signature identifies the specific intrusion. Knowledge of the specific type of attack means that an appropriate response can be determined immediately. For its accuracy, misuse detection is widely preferred in commercial systems. However, misuse detection is not able to detect new and novel attacks. And new signature must be developed whenever a new attack is discovered.

Anomaly detection tries to characterize normal behavior, and everything else is assumed to be anomalous (although not necessarily malicious). The underlying premise is that malicious activities will deviate significantly from normal behavior. The characterization of normal behavior is called a normal profile which is usually constructed by statistical analysis of training data obtained from observations of past normal behavior. A major advantage of anomaly detection is the potential to detect new attacks without prior experience. At the same time, it suffers from high false alarm rate.

In specification-based detection [72], the correct behaviors of critical objects are abstracted and crafted as security specification, which are compared to the actual behavior of the objects. Intrusions, which usually cause an object to behave in an incorrect manner, can be detected without exact knowledge about the nature of the attacks.

Based on the network architecture, IDSs can be divided into two types. One type of intrusion detections falls into a single layer design which normally involves the information from

the network layer. The other type of intrusion detection techniques involve cross layer design [49, 55, 69, 70, 85]. The behavior from multiple layers in the network stack have been observed and analyzed for intrusion detection.

2.2.2 Single-layer Based Intrusion Detection Methods

Watchdog, introduced by Marti et al. [53] was the first snooping ID protocol for MANETs. *Watchdog* relies upon Dynamic Source Routing (DSR) [44] and each node participates by "watching" its downstream node, on the route from source to desitnation, to ensure that it has re-transmitted the packet without modification. The authors asserted hold that if source routing is not used then a misbehaving node could simply broadcast to a non-existent node to fool the watchdog. While this is true, packet modification is not covered up by simply broadcasting to a non-existent node. To mitigate the effects of a misbehaving node, the authors introduce *Pathrater*, which selects a path from source to destination based upon a "reliability" metric, instead of the shortest path. This approach relieves the malicious node from the requirement of participating in the routing process, which may be construed as a reward.

Buchegger and Le Boudec [28] build upon Marti et al.'s work by replacing *Watchdog* with *Neighborhood Watch*, which is also dependent upon DSR, and snoops its downstream neighbor. They introduce a *Trust Manager*, *Reputation System*, and a *Path Manager*. Essentially each node is required to run a finite state machine to calculate trust, which in turn is used to decide the other node's reputation and then to determine routes with the highest security metric.

Tseng et al. [72] present a specification based IDSs over the Ad hoc On-Demand Distance Vector (AODV) routing protocol. The IDS is built on a distributed network monitor architecture that traces AODV request-reply flows. Network monitors audit every route request (RREQ), route reply (RREP) and route error (RERR) in order to build and update complete request-reply session trees and corresponding forwarding tables. Constraints on the request-

reply flow are specified using finite state machines. Once the correct behaviors of AODV are manually abstracted and crafted as security specifications, and this is compared with the actual behavior of the AODV. Intrusions, which usually cause object to behavior in an incorrect manner, can be detected without exact knowledge about them.

Patwardhan et al. [56] propose an approach of securing a MANET using a threshold-based intrusion detection and a secure routing protocol. In their work, a two-pronged approach for protecting MANETs against attacks - secure the routing process and deploy IDSs on individual nodes throughout the network to detect misbehaviors. The secure routing protocol is based on IPv6 and the IDSs is based on "watchdog" mechanisms. That is, a common neighbor of two nodes on a link will be selected to watch packet transmission. If packet dropping rate reaches a certain threshold, an alarm will be triggered at the monitoring node.

2.2.3 Cross-layer Based Intrusion Detection Schemes

Zhang et al [85] propose the idea of multi-layer intrusion detection in wireless ad hoc networks. They present a cooperative distributed intrusion detection and response framework for MANET. Anomaly detection is the primary ID approach discussed, including anomalies in routing updates, abnormalities at the MAC layer (number of channel requests, etc.)

[70] presents a cross-layer design for DoS attacks. Two different ways for cross layer detection have been provided: CIDS-I in which detection information at one layer triggers other layers detection in the protocol stack; CIDS-II includes a detection that is based on information collected from different layers.

In [49], Liu et al. propose a node-based intrusion detection system (IDS) for wireless ad hoc networks. They define a feature set that correlates information from MAC layer and network layer to profile normal behaviors of mobile nodes, and they also adapt a rule-based data mining technique for anomaly detection.

[69] utilizes cross layer information for jamming attacks detection in wireless ad hoc net-

works. Specifically, they they provide a monitor based intrusion detection system for wireless ad hoc networks. They model the jamming attacks at different layers of the protocol stack and studied the effects of different jamming attacks on the network performance. A cross-layer approach is adopted to estimate the network congestion in order to reliably evaluate the presence of jamming.

2.3 Intrusion Response in Multi-hop Wireless Networks

2.3.1 Introduction

Intrusion response is generally referred to a series of actions that can be deployed to thwart attack and ensure the safety of computer systems. It is often integrated with IDS and triggered when IDS has detected intrusion and raised the alarm. However, intrusion response has not gained as much attention as intrusion detection in both industry and academia. In most existing security systems, it highly relies on the expertise of system administrators to select or deploy response actions. There is a need for IRS and IDS to cooperate and work in parallel as attackers intervene in an automated way, at computer speed.

In wireless mesh networks, there are only a handful of research work [18, 75] that have conducted in intrusion response. Unlike wired network, wireless mesh networks are more vulnerable to malicious attacks, have distributed network architectures, and present dynamic network topology. Those characteristics may require different response actions and strategies.

2.3.2 Existing Solutions for Intrusion Response

Stakhanova et al. [63] have provided a general overview of existing work in intrusion response system. Based on triggered response activities, an IRS could be a passive system which mainly generates notification and provides response information. On the contrary, an active response system aims to minimize damage cost and trace attackers. Based on degree

of automation, IRSs can be divided into notification system, manual response system, and automatic system. Today, most of the research in IRS focus on automatic response systems.

Lee et al. [47] proposed a cost-sensitive model for intrusion detection and response. Three cost factors are identifed: *operational cost* that covers the cost of processing and analyzing data for detecting the intrusion, *damage cost* which measures the cost caused by the attack and *response cost* that includes the operational cost deploying the response actions. The three cost facts act as metrics for selecting intrusion response measures. Starting with a taxonomy of attacks that have been given by a reference dataset, empirical costs for the attack damage and reactions have been defined.

Toth and Kruegel [71] represents a network system in a dependency tree which models the configuration of resources (network services provided by hosts), users, network topology, and firewall rules. Different dependency relation between resources and users are shown in the model. Once an intrusion is identified and a response action is added into the model, the penalty cost caused by the response to the system is evaluated. The one with the least penalty cost to the system is selected. The model is used to select a globally optimal response.

Balepin et al. [22] use a direct graph to model local resources and their dependencies. Nodes in the graph represent specific resources in the system and edges represent dependencies between them. The authors propose to use costs of priority resources as base metric for response choice. In their system map, only priority nodes - representing the important system resources - have a cost value of their own. Cost values are assigned to other nodes based upon the fact, that priority nodes depend on them. The cost values are set by the network administrator on creation of the system map. Subsequently, the following values are computed: Intrusion Damage (sum of all cost values of the affected nodes), Response Benefit (sum of all cost values of the nodes, that are restored to safe state by the response), and Response Costs (sum of all cost values of the nodes, that are negatively affected by the response).

ADEPTS [81] is a more complex framework for determining automated responses against attacks. In the framework, two types of graphs are used: a server graph (S-Graph) which rep-

resents the inter-dependencies between services and an attack graph(I-Graph) which describes the attack state and their possibilities. I-Graph is created based on S-Graph and later the attack graph is used for response selection. The responses are selected based on the effectiveness of this response to a particular attack in the past. To determine when to deploy the response ADEPTS uses *Compromised Confidence Index (CCI)* which expresses the probability that the goal represented as a node in I-Graph is currently achieved by the attacker.

Jahnke et al. [43] generate a dependency graph for system resources in which nodes of the graph represent resources and edges represent different types of dependencies among resource entities. This dependency graph is used for both quantifying the effects of a response measure after its application and for choosing the most promising alternative of a set of available response measures.

2.4 Summary

In this chapter, we have presented background and related work in security research over wireless mesh networks. Especially, we focused our literature review in group key management, intrusion detection and intrusion response. In next chapters, we will introduce our new schemes of a heterogeneous group key management, cross-layer based intrusion detection, and a generic intrusion response model.

CHAPTER 3. A HETEROGENEITY-AWARE FRAMEWORK OF GROUP KEY MANAGEMENT IN WIRELESS MESH NETWORKS

3.1 Introduction

Group key management plays an essential role in securing multicast communication for WMNs. The problem has been extensively studied for both wired and wireless networks, which result in three typical categories of solutions [58]: centralized [29, 30, 73, 79], decentralized [26, 35, 54, 59, 62], and distributed [45, 57] key updating protocols. The centralized method relies on a trusted third party called group server to generate and distribute group keys. With this method, the logic tree-based schemes proposed in [21, 74, 79] are representative. Although their communication, storage and computation cost is $\Theta(logn)$, where n is the group size, the communication cost and the rekeying delay are still high when these schemes are applied to a large scale network. The decentralized method distributes the group management duty to multiple subgroup leaders in order to reduce the load at a single point. In a distributed key management scheme, there is no explicit group server, and keys are generated collaboratively by one or multiple group members. Many group key management techniques for wireless ad hoc networks and sensor networks fall into the third category. Particularly, these schemes may take the threshold-based approach, in which the group key is either agreed among group members or generated based on shares from group members [48, 51, 83].

However, none of existing solutions fits well in WMNs, since they cannot take advantage of or mitigate the disadvantage of the architectural heterogeneity of WMNs. Unlike wireless ad hoc network or sensor networks which are solely composed of homogeneous wireless

devices, WMNs are heterogeneous. In such networks, different types of networking components are integrated which requires our security design to accommodate different requirements and/or limitations. For instance, mesh routers and mesh clients have different power constraints: mesh clients usually have constrained power and energy supply while mesh routers do not. In addition, conventional wireless networks provide a wired infrastructure and have only the wireless connectivity at the last mile, WMNs provide a multi-hop wireless infrastructure which lacks physical protection. The nature of wireless communication within WMNs makes it highly vulnerable to security attacks [15, 61]. Furthermore, the hierarchical network structure of WMNs can be exploited to facilitate group key management. In [66], network topology is leveraged in building the key tree over cellular network. However, this method cannot be directly applied to WMNs since the topology considered by them has only one-hop wireless communication.

We propose a novel heterogeneity-aware group key management framework which combines the logical key hierarchical technique together with the localized threshold-based technique. Specifically, a WMN multicast session involves both static backbone nodes and mobile client nodes: for backbone nodes which are topologically-stable but may spread over a large area, the localized threshold-based group key management technique is applied since the technique requires stable network topology and its rekeying delay is independent of network scale; for client nodes associated with each backbone node which are mobile but confined in a limited area, the logical key hierarchical technique is applied since the technique is resilient to node mobility but its rekeying delay is proportional to the network scale. The contributions of this work include, but not limited to:

- *Advantages of heterogeneity of WMNs:* Our framework take the advantage of heterogeneity of WMNs and divides the group key management into two layers. The top tier includes all backbone nodes and adapts a threshold based group key agreement protocol to update shared session keys. The bottom tier is composed of a backbone node and the

associated group members. Each backbone node provides the subgroup leader capability which includes building and maintaining a logical key hierarchy (LKH) [73, 79] for corresponding group members.

- *Localization of key refreshment:* The key refreshing for both the top layer and bottom layer happen at the local environment.

- *Resilience to both insider attacks and outsider attacks:* The property of the threshold-based group key agreement protocol over the wireless mesh backbone ensure our framework resilient to both insider attacks and outsider attacks.

- *Reduction of rekeying delay and storage overhead at end nodes:* Comparing with traditional LKH where a single key tree is built for the group and new keys have to be sent by the group server, our framework localizes the key refreshing at the subgroup leader and a single hop communication might be the only requirement for key updating. In addition, the key tree at the mesh router can be much smaller. Therefore the number of keys required for a group member can be much smaller.

The chapter is organized as follows. Section 3.2 describes the system model on which our group key management is based. Section 3.3 overviews the heterogeneity-aware framework, identifies technique problems in the integration and lists possible solutions. Section 3.4 presents a specific implementation of group key management system under our framework. Section 3.5 provides analysis of security, communication load and other related issues. Performance evaluation results are shown in section 3.6. Finally, section 3.7 concludes the work.

3.2 System Model

3.2.1 Trust Model

In our design, we assume that a group server is a trusted entity resident in the network and it is responsible for group initialization and information collection from mesh routers. The group server publishes its public key into the network and a subgroup leader can generate and distribute keys. We also assume that a mesh router in a WMN can conduct authentication activities. For instance, Zhang et al [84] provides a security architecture for wireless mesh client and mesh routers to conduct mutual authentication. Based on those security mechanisms we assume that the group server can trust the mesh routers during group initialization. We later release this assumption in our scheme.

3.2.2 Attack Model

In this section, we divide attacks against the WMNs into two categories: outsider attacks and insider attacks. Outsider attacks usually do not jeopardize the network. They commonly do network monitoring and traffic analysis on the target network.*Eavesdropping* and *traffic analysis* are typical outsider attacks. On the contrary, insider attacks directly target at breaking down the networks. *Router failure*, *message compromising* and *collusion attacks* all belong to this category. In the following, we list those attacks:

- *Eavesdropping* Eavesdropping attack is one type of outsider attacks. It occurs when someone listens to or eavesdrops on network traffic. The eavesdropper can capture and analyze message exchange in order to reveal the secret keys. Eavesdropping attacks are by their very nature difficult to detect.

- *Traffic analysis* Another type of outsider attacks is traffic analysis. Unlike an eavesdropping attack, an adversary in traffic analysis may explore the vulnerability of cryp-

tography method to crack the group keys. This type of attack is out of the scope of this work.

- *Router failure* The purpose of this type of insider attack is to physically break down systems in order to breach the network service. For instance, an attacker may crack the system running on the router in order to make the broadband service unavailable to clients. In other cases, an adversary can physically capture the node [67] and crack the security information on the node. Those attacks will crash mesh routers in the mesh network, thus fail the service provided to clients. A direct consequence of such an attack is that those clients associated with the victim router will lose the connection. In order to handle such an accident, redundancy or other security mechanisms have to be provided.

- *Message compromising* Message compromising is another type of insider attack as the attacker intends to reveal secret information of the network and steal useful messages or data from the network. By compromising the system running on a mesh router, an intruder can infer secret keys and be able to decrypt the message shared only by group members. In some situation, this attack is the extension of the *router failure* attacks. For instance, an adversary can physically capture the node and crack the security information on the node. Then the adversary places the compromised node back on the network and this node will resume all the responsibilities of a legal node. In such a way the adversary will be able to get all useful information from the compromised node.

- *Collusion attack* Collusion attack is a type of insider attack that could happen among several colluding mesh routers. Routers that are compromised can build secret tunnels between one another and use them to exchange messages. Once enough information is gathered, it can be used to crack the group keys of the network.

3.3 A Heterogeneity-Aware Group Key Management Framework

We propose a complete framework for group key management in WMNs. The framework takes advantage of the heterogeneity of WMN and applies both tree based hierarchical techniques and threshold-based techniques.

In general, our framework divides the key management into two layers: bottom layer and top layer. The bottom layer includes a set of subgroups. Each subgroup is composed of a mesh router and a set of group members associated with it. The mesh router will act as the subgroup leader. In each subgroup, the subgroup leader builds and maintains a logical hierarchical key tree for its associated members. A client joins/leaves the group through the subgroup leader which is responsible for membership verification and key updating. A mesh router becomes a subgroup leader after it is authenticated by the group server. The root key of the key tree at the subgroup is called **sub-key**.

At the top layer, all subgroup leaders form a wireless ad hoc network where a threshold-based key management technique will be applied. We do not restrict any particular technique at this point as many can be adopted. For instance, Asokan et al in [19] proposed a password-based multi-party Diffie-Hellman group key agreement protocol which is mainly for a physically presented group that shares a password, e.g. groups in a meeting room or a classroom. Some other threshold-based group key management protocols allows group members to compute the group key based on their individual contributions providing verifiable trust relationship between participants [48, 51, 83]. In our framework implementation, the threshold-based group key management in [83] is applied. We call the shared key among subgroup leaders a **session-key**. This session key is also held by the group server.

In the following, section 3.3.1 presents a multicast session in a WMN. Section 3.3.2 describes subgroup leader initialization. The rekeying process for a group member join/leave is shown in section 3.3.3, and section 3.3.4 discusses revocation of a subgroup leader.

3.3.1 A Multicast Session in WMN

Our framework supports both one-to-many and many-to-many [42] group communication. In one-to-many group communication, the message flows from the group server to group members; while in many-to-many group communication, each member can be both sender and receiver.

We first describe the process involved in a one-to-many multicast session: the message will be first encrypted using the shared **session-key** on the top layer, then be disseminated to the network. Once a subgroup leader receives the message, it first decrypts the message and then encrypts the message using the **sub-key** shared by all subgroup members in the bottom layer. After that, the message is broadcast by the leader. All clients in the transmission range may receive the message. But only those clients that have joined the group will be able to decrypt the message. The process is also shown in Fig. 4.1:

(a) GS $\rightarrow SL_i$: $K_t\{M\}$, $i = 1, ..., n$

(b) SL_i: $K_t^{-1}\{K_t\{M\}\}$

(c) $SL_i \rightarrow GM_{i_j}$: $K_b\{M\}$, $j = 1, ..., m$

(d) GM_{i_j}: $K_b^{-1}\{K_b\{M\}\}$

Here GS is the group server, M represents the message, SL_i ($i = 1, ..., n$) is the i_{th} subgroup leader, GM_{i_j} ($j = 1, ..., m$) is the j_{th} group member of SL_i, and K_t and K_b denote top layer **session-key** and bottom layer **sub-key**, respectively.

In many-to-many group communication, group members can share messages among one another. In this case, a group member is the sender and other group members are receivers. The sender first encrypts the message using the subgroup key **sub-key** and sends it to the corresponding subgroup leader. The subgroup leader then is required to process the message such that its associated group members can receive it and group members associated with

other subgroups can also receive it. To accomplish the first goal, the subgroup leader simply rebroadcast the message to its own subgroup. Then the subgroup leader decrypts the message and encrypts it using the top layer **session-key** and broadcasts it to backbone. Other subgroup leaders conduct the same process as in one-to-many multicast session. After that, the encrypted message will be distributed to the associated group members. A many-to-many multicast session is shown in Fig 4.1:

(1) $GM_{i_j} \rightarrow SL_i$: $K_b\{M\}$

(2) $SL_i \rightarrow GM_{i_k}$: $K_b\{M\}$, $k = 1, ..., m$

(3) GM_{i_k}: $K_b^{-1}\{K_b\{M\}\}$, $k \neq j$

(4) SL_j: $K_b^{-1}\{K_b\{M\}\}$, $k \neq j$

(5) $SL_j \rightarrow SL_l$: $K_t\{M\}$, $l \neq j$

(6) SL_l: $K_t^{-1}\{K_t\{M\}\}$

(7) $SL_l \rightarrow GM_{l_j}$: $K_b\{M\}$, $j = 1, ..., m$

(8) GM_{l_j}: $K_b^{-1}\{K_b\{M\}\}$

3.3.2 Initialization of Subgroup Leaders

In the previous section, we do not differentiate subgroup leaders and mesh routers as we assume that each mesh router acts as a subgroup leader. However, a mesh router may not have any member nodes connected to it. In this case, including mesh nodes that do not participate in the group is not necessary. We can add a mesh router into the group when there is a demand. We call the latter method *reactive initialization* and the former one *proactive initialization*.

Proactive initialization configures all mesh nodes in a WMN as subgroup leaders during group initialization. In this method a new node can join the group without any delay. The drawback is that **session-key** is widely spread and is more likely to get compromised.

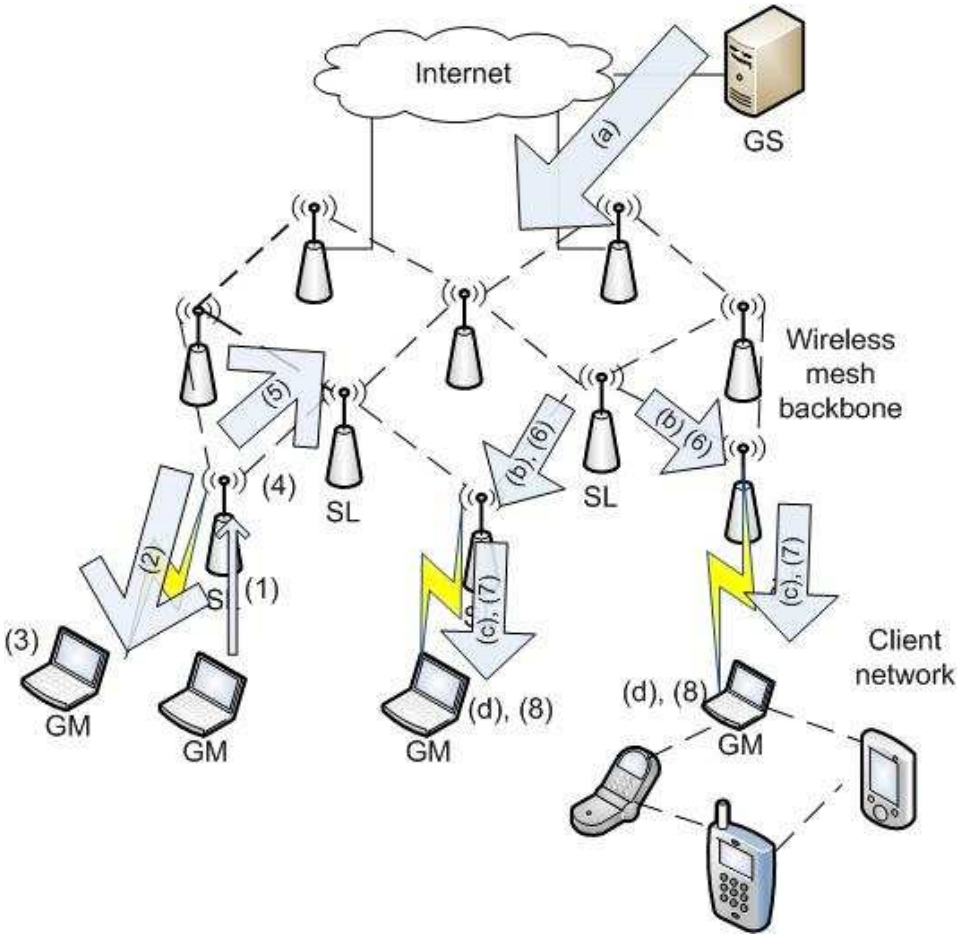


Figure 3.1 WMN multicast session (GS: group server, GM: group member, SL: subgroup leader, one-to-many multicast: (a)-(d), many-to-many multicast: (1)-(8))

In a *reactive* method, a mesh router is initialized as a subgroup leader when its first subgroup member joins the group. It has the drawback of delaying a new member's join as it has to yield the corresponding mesh router to get initialized from the group server.

Another issue related to subgroup leader initialization is how to authenticate a group member. A subgroup leader can forward such a join request to the group server and waits for a positive reply to add the member to its subgroup. In this case, the group server will be frequently invoked during the group communication. A more efficient way is to involve subgroup leaders in membership verification. However, subgroup leaders can not be fully trusted by a group

member as they could get compromised. In 3.4.1.1 we propose a Bloom Filter [25] based authentication method to allow a subgroup leader to verify a client's membership without any complementary secure information.

3.3.3 Join/Leave of a Group Member

In a WMN, a group member may join or leave the group at any time. To ensure forward security, when a group member leaves, the related subgroup leader deletes corresponding key information from the key tree and updates **sub-key** as well as related key encryption keys (KEKs) [74] held by this departing node.

Similarly, backward security has to be ensured for a join request from a group member. The associated subgroup leader generates a shared key for a new member and inserts it into the key tree as a leaf node. Then all corresponding KEKs from the leave node to the root node will be updated and sent to the new node.

A join/leave request can also be generated during node mobility. When a group member moves from place to place, its subgroup membership will may changed. In order to continue an on-going multicast session, a mobile member needs to join the new subgroup. In addition, as it leaves the old subgroup, key refreshment has to be invoked in the old subgroup. A mobility management scheme in will be provided in section 3.4.4 to ensure seamless handover in our framework.

3.3.4 Revocation of a Subgroup Leader

A subgroup leader will be revoked if it is detected behaving maliciously. The first step for the revocation at every other subgroup is to reset timer and launch the **session-key** updating process. We assume a backup router is installed or coverage redundancy is provided in the wireless mesh backbone. A backup router replicates the original mesh node. Therefore, the switch will be transparent to a client. For coverage redundancy, a client node is covered by at

least two mesh routers. Then the neighbor nodes of the revoked subgroup leader will send out messages to notify those group members. A group member can respond to such a message to join a new subgroup.

3.4 A Heterogeneity-Aware Group Key (HAGK) Management Scheme

In the above section, we have presented a heterogeneity-aware framework for group key management in WMNs. In this section, we introduce a specific group key management scheme to implement the framework, namely, a heterogeneity-aware group key (HAGK) management scheme. In this scheme, two techniques are adopted into the framework: the threshold-based group key management in [83] is applied to the top layer and the traditional logical key hierarchy (LKH) [79] is used in the bottom layer.

The HAGK key management scheme includes three components: group initialization, threshold-based key agreement protocol and logical key hierarchy. We describe each component in the following.

3.4.1 Group Initialization

During group initialization, subgroup leaders are configured appropriately such that they can authenticate a group member. They are also loaded with necessary information for the upper layer threshold-based group key management. These two tasks are completed during *subgroup leader initialization* and *key pre-distribution*.

3.4.1.1 Subgroup Leader Initialization

In our framework, a subgroup leader participates the top layer group key agreement and is also responsible for maintaining a key tree for connected members. In addition, it is able to verify a client's membership. In the following, we propose a Bloom Filter [25] based authentication method, called *semi-anonymity authentication*, to allow group members to get

authenticated without revealing their secure information to a subgroup leader. It is semi-anonymous as the subgroup leader requires the identification of a group member. To ease description, we assume a *proactive initialization* is used. We will explain how this method can be extended to *reactive initialization* in section 3.4.5.

Bloom Filter Bloom filter is widely used for membership query. A Bloom filter represents n elements in a set $B = b_1, b_2, ..., b_n$. It includes a vector v of m bits, initially all set to 0, and k hash functions $h_1, h_2, ...h_k$, each with hash value in $1, ...m$. For each element $b \in B$, the bits at position $h_1(b), h_2(b), ..., h_k(b)$ in v will be set to 1. Given a membership query for c, if any of the bits at $h_1(c), h_2(c), ..., h_k(c)$ is 0, then certainly c is not in the set B. Otherwise, we conjecture that c is in the set. However, there is a certain probability of false positive.

The salient feature of Bloom filter is that there is a clear tradeoff between m and the probability of a false positive. Observer that after n keys into a table of size m, the probability that a particular bit is still 0 is exactly

$$(1 - (\frac{1}{m}))^{kn}$$

. Hence the probability of a false positive in the situation is

$$(1 - (1 - (\frac{1}{m}))^{kn})^k \approx (1 - e^{\frac{kn}{m}})^k$$

The right hand is minimized for $k = \ln 2 \times m/n$, in which case it becomes

$$(\frac{1}{2})^k = (0.6185)^{m/n}$$

.

The subgroup leader initialization includes the following steps:

1. Each client is required to register with the group server before the group communication starts. The group server will generate a unique ticket for each client. A ticket can be the hash value over user identification and password or a type of certificate.

$$GS \rightarrow GM_i : T_i, i \in 1, ..., n$$

Where T_i is the ticket for GM_i.

2. The group server stores each ticket in its storage and generates a Bloom Filter for all tickets $T_i, i \in 1, ..., n$.

$$GS : generate \quad BF$$

3. Before the group communication starts, the group server broadcast the Bloom Filter BF to the network. To ensure authenticity, the group server can encrypt BF using its private key.

$$GS \rightarrow SL_j : k_{ps}\{BF\}$$

where k_{ps} is the private key of the group server.

4. Upon receiving the Bloom Filter BF from the group server, a mesh node decrypts it using the group server's public key. Then the mesh nodes generates a secret key k_i and sends k_i to the group server together with its identification.

$$SL_i \rightarrow GS : k_{pb}\{id, k_i\}$$

k_{pb} is the public key of the group server

When a registered group member wants to join the group communication, it sends its identification together with its ticket T_i to the associated mesh node. Upon receiving T_i, the subgroup leader verifies its membership based on the saved Bloom Filter.

In the above authentication method, a malicious mesh router could change the Bloom Filter it saved to disallow a legitimate member node and generates any ticket for a malicious client. We will address this issue in section 3.5.3.

For a group with n group members, m and k have to be selected in the Bloom Filter generation to ensure low false positive rate and also low communication overhead. For instance, in a large group with 1024 nodes, if we want to ensure a false positive rate of 0.01, m can be set

to $10,240$ with $k = 5$. A lower false positive rate of 0.001 can be ensured if m is set to $15,360$ bits and $k = 8$. In both cases, the size of m is about 2 packet size if each packet is 1000 bytes.

false positive rate	k	m (bits)
0.01	5	10,240
0.001	8	15,360

3.4.1.2 Key-predistribution

At the end of *subgroup leader initialization*, each mesh router will share a session key k_i with the group server. Then the group server will use this session key for key-predistribution as specified in [83].

The group server constructs a t-degree (t is a system parameter) univariate polynomial $p(x)$ over a prime finite field $F(q)$ to represent the keys of the group, where $p(0)$ is the initial group key, $p(c)$ ($c \geq 1$) is the group key of version c, and q is a large prime whose size can accommodate a group key. The group server then sends out the group polynomial $p(x)$ to every mesh router in the wireless mesh network using the received secure key.

Upon receiving the group polynomial $p(x)$, every mesh router u will

- randomly pick up a polynomial, called *encryption polynomial*, to encrypt the group polynomial $p(x)$. The encrypted group polynomial is $p'(x)$;
- calculate a share for each member in its neighbor list using the encryption polynomial;
- send out the share to each corresponding neighbor and delete the original group polynomial $p(x)$.

The encryption polynomial is a bivariate polynomial. For instance, the randomly-picked encryption polynomial at mesh router u is as follows:

$$r_u(x, y) = \sum_{0 \leq i \leq t, 0 \leq j \leq \mu} A_{i,j} x^i y^j,$$

where t and μ are system parameters that can be adjusted to satisfy the requirement of the security mechanism.

Using the encryption polynomial $r_u(x, y)$, the mesh router u encrypts the group polynomial $p(x)$:

$$p'(x) = p(x) + r_u(x, u)$$

We assume each router u maintains a neighbor list that includes all the one-hop neighbors and two-hop neighbors.

In this paper, we will use neighbor and one-hop neighbor interchangeably and we will specify two-hop neighbor when needed. The major purpose to maintain two level neighborhood information at each router is to get enough neighbors to satisfy the security requirement of the system which will be explained in section 3.5.

After the group polynomial encryption, mesh router u distributes a share of $r_u(x, y)$ to each member in its neighbor list. Specifically, every member v_i in the neighbor list gets a share of $r_u(x, v_i)$. At the same time, u removes the encryption polynomial $r_u(x, y)$ and the group polynomial $p(x)$, but keeps $p'(x)$ and the initial group key $p(0)$.

3.4.2 Threshold-based Key Agreement Protocol

Within the wireless mesh backbone of wireless mesh network, a threshold based key agreement protocol [83] is employed to periodically update the **session key** shared by all subgroup leaders.

Each subgroup leader maintains a *rekeying timer*, which is used to periodically notify the subgroup leader to update its group key, and the current version of the group key (denoted as c). Note that c is initialized to 0 when the group server initializes the group setting.

To update group keys, each mesh router u increases the value of c by one, and returns share $r_{v_i}(c, u)$ to mesh router v_i. v_i could be either u's one-hop neighbor or two-hop neighbor. Meanwhile, router u receives a share $r_u(c, v_i)$ from mesh router v_i. Having received $\mu + 1$

shares from its neighbors and two-hop neighbors, which can be denoted as $< v_i, r_u(c, v_i) >, i = 0, ..., \mu$, router u can reconstruct a unique μ-degree polynomial

$$r_u(c, y) = \sum_{j=0}^{\mu} B_j y^j$$

by solving the following $\mu + 1$ $(\mu + 1)$-variable linear equations:

$$\sum_{j=0}^{\mu} (v_i)^j B_j = r_u(c, v_i), i = 0, ..., \mu.$$

By knowing $r_u(c, x)$, router u can compute its new group key $p(c) = p'(c) - r_u(c, u)$.

3.4.3 Hierarchical Key Tree

Each mesh router acts as a subgroup leader which builds and maintains a key tree for all associated group members. Upon receiving a join request from a new member, the mesh router does not forward the join request to the group server as it does in a traditional hierarchical key tree management. Instead, it generates a pairwise key for the new client and returns the corresponding KEKs in the key tree to the new client. Similarly, when a member leaves (or is evicted from) the group, the subgroup leader will update all the keys held by the departing member. That is, all the KEKs in the path from the parent of the departing member to the root will be regenerated and distributed into the local client network. More detailed operations of member join/leave can be found in [74] and [79].

Meanwhile, the router keeps a record of its group members which may include the member id number, joining time, and departing time. All the information will be periodically reported to the group server to maintain the log.

3.4.4 Mobility Management

During a group communication over a WMN, great care must be taken to ensure seamless handover when a group member moves from place to place and its subgroup gets changed. Our framework provides a simple algorithm mobility management.

We use an example to show the mobility management in our scheme. In Fig. 3.2 a group member GM moves from $placeA$ to $placeB$, to $placeC$, and to $placeD$ in which it, respectively, associated with mesh router 3, 5, 6 and 9. GM leaves the group at $placeD$ where it is associated with mesh node 9. We denote the subgroup at each mesh router as SG_i $(i = 1, ..., 9)$. The steps involved in mobility management are stated as follows.

1. GM initially belongs to SG_3. When GM is about to handover from SG_3 to SG_5, it sends SG_3 a "roam" message together with a random number r. Then SG_3 issues a *transfer ticket*: $h(GM, k_s, r)$ to GM where GM represents the member id, k_s is the current $session - key$ in the top layer. At the same time, SG_3 broadcast a message to its neighbors (e.g. SG_1, SG_2, SG_4 and SG_5) with the id of the roaming group member. SG_3 then inserts the id of GM into its *roam_list* which includes the ids of group members that roamed out of this subgroup. Note that the old subgroup leader SG_3 does not unicast the message as it does not know to which neighbor the group member is moving to.

2. From the first step, SG_1, SG_2, SG_4 and SG_5 cache GM's id in a list called *waiting_list*. Once GM selects to join the subgroup SG_5, it sends its identification and the random number r to SG_5. SG_5 checks its *waiting_list* and gets to know that GM comes from subgroup SG_3. SG_5 then replies with a hash value $h(GM, k_s, r)$ and the corresponding keys from the key tree to GM as if a new group member joins the subgroup. Upon receiving the reply from SG_5, GM verifies the received value from SG_5 equal to its *transfer ticket*, which shows that SG_5 is still an active subgroup leader in the group communication. GM then accepts keys from SG_5. Otherwise, it selects another subgroup leader. Once SG_5 gets verified, GM replies SG_5 an "accept" message. SG_5 then caches a reverse path to SG_3 in its *roaming_list*. Other neighbors will delete the id from its *waiting_list* if no association is received after a certain time.

3. When GM leaves SG_5 and joins SG_6, SG_5 will act the same way as SG_3 and SG_6 will conduct all the actions in the second step. So does the handover between SG_6 and SG_9.

4. Eventually, a group member will leave the group. As in the example, GM leaves the group from SG_9. SG_9 will first update its key tree after GM sends the leaving request. Then SG_9 will check its *roaming_list* and sends a "Roaming Member Leave" message to SG_6 according to the reverse path.

5. Upon receiving "Roaming Member Leave" message, SG_6 deletes GM from its *roaming_list* and updates its key tree as if a group member leaves the subgroup. Similar to SG_9, SG_6 forwards the message to SG_5, then SG_5 to SG_3.

6. SG_3 updates its *roaming_list* and key tree accordingly. No message will be sent out from SG_3 as SG_3 is the last hop along the reverse path.

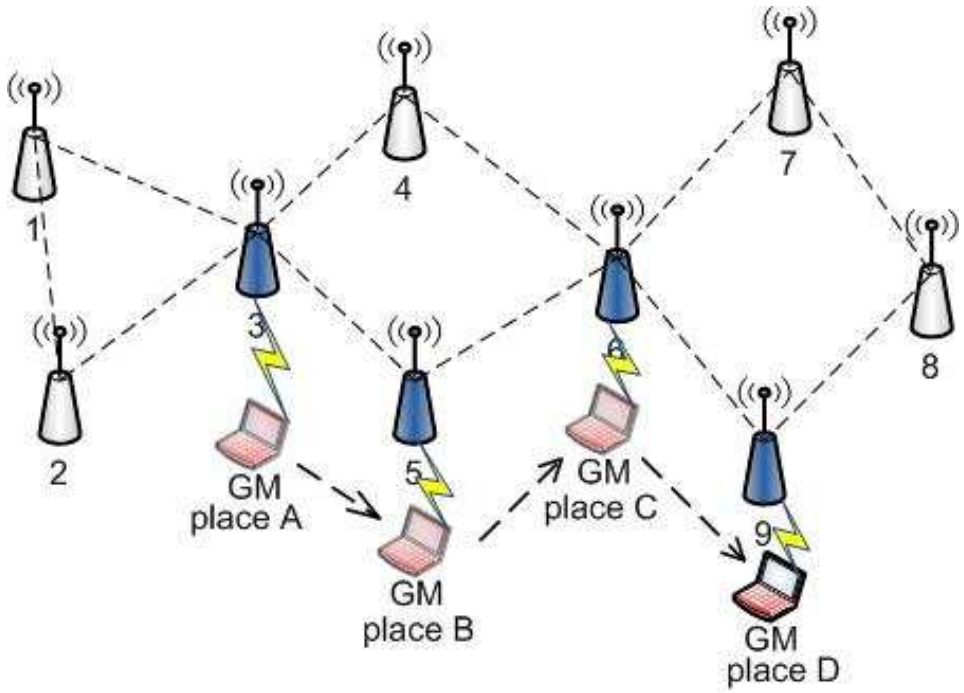


Figure 3.2 Mobility management of group key management in WMNs

3.4.5 Reactive Initialization

The threshold based group key agreement protocol requires each subgroup leader has a certain number of neighbors. Under this condition, *proactive* method is more appropriate. If *reactive* method is applied, we allow those subgroup leaders to establish an overlay network above the wireless mesh backbone. And for a small size of overlay, our threshold based key agreement protocol degrade to Diffie-Hellman method where all subgroup leaders contribute to agree on a **session-key**.

3.5 Discussion

3.5.1 Security Analysis

In section 3.2.2, several attacks in WMNs have been described. In the following, we'll show that our HAGk group key management is resilient to those attacks or makes them hard.

Outsider attacks won't be useful in revealing the two shared keys: **session-key** and **sub-key**, in HAGK group key management. For instance, for an attack that aims to infer the group key polynomial at the top layer, the attacker needs to monitor several links at the same time. It is first required to get $r_u(c, v_i)$ which is encrypted using the secret key shared between backbone nodes u and v_i. Even ($\mu + 1$) such shares are retrieved, the passive attack still can not infer the original group key polynomial $p(x)$ as $p'(x)$ is unknown which is kept by the backbone node and is never been transmitted in the network.

When a new backbone is added into the network, $p(x)$ is transmitted to the new subgroup leader. An adversary may monitor the traffic in the network at this time in order to get $p(x)$. As the network topology of WMNs is relative stable, the chance that a new added backbone will also join an on-going group communication is rare.

For the *router failure* attack, if the wireless infrastructure provides such a redundancy for the client network, then the problem is solved. With the redundancy deployment, each area

is cover by at least two backbone nodes. If one fails, the other one can be used. Those disconnected group members can rejoin the group though the other backbone node. If no redundancy is guaranteed at the wireless infrastructure, the group server or the neighbor node for the cracked backbone node will cache missed messages for group members that are disconnected from the group. When they get back to the group later, the cached message will be resent to the subgroup.

The HAGk group key management is resilient to collusion attacks up to size of $\mu+1$. This is decided by the threshold-based key updating protocol at the top layer. A new **session key** is calculated at each subgroup leader based on the $\mu+1$ contributions from its one-hop and two-hop neighbors. According to [83], the group polynomial p(x) of group g is compromised if and only if (1) a subgroup leader u is compromised and (2) at least $\mu+1$ neighbors (including two-hop neighbors) of u are compromised or at least $t+1$ past session keys of the group are compromised. Therefore, our group key management is immune to collusion attack of size up to μ. The detailed proof of these properties can be found in [83].

For *message compromising* attack, an hacker can compromise one mesh router in order to obtain group polynomial $p(x)$ that can generate all the **session-keys** shared by all mesh routers. Physical capture attack may only reveal $p'(x)$ on the node because $p(x)$ is not saved. Then an attacker can adopt other hacking methods to compromise the system on the router. The hacker needs to be able to get $r_u(c, v_i)$ before cracking the group key. This process might be risky as deviated behaviors could be observed by an intrusion detection system deployed in the network.

3.5.2 Communication Overhead

We further analyze the communication overhead caused by the threshold based group key agreement protocol within subgroup leaders. In each **session key** refreshing cycle, every subgroup leader is required to calculate and send out a share to each of its one-hop neighbors

and two-hop neighbors. We assume the degree of each mesh router is d, then the communication has the complexity of $O(d^2)$. The worst case is when the backbone is fully connected, $d = n$, where n is the number of mesh routers. To reduce the communication load for each mesh router, an optimization mechanism can be applied. That is, the rekeying message can be piggybacked with the routing messages. For instance, some routing protocols requires a periodical exchange of neighborhood information. We can set the **session key** refreshing timer equal to the exchange timer in the routing protocol, then the shares that are used for key updating can be combined with the exchanging messages.

3.5.3 Other Issues

In the proposed scheme, we do not specify whether a group member is directly connected to the wireless mesh network or is connected through a client network. Our method can be applied in both situations. In the latter situation, two cases will be involved. In one case all clients in the client network is included in the multicast group. Then the node that bridges the client network and the mesh router can be treated as a single ”member” in the subgroup. Once the message reaches the bridging node the message will be distributed following the group communication mechanisms agreed within the local client network. In the other case, not all clients in the client network join the group. The corresponding subgroup leader will treat each client as if it is connected to the mesh network directly.

Furthermore, our method assumes that during the predeployment process the mesh routers can be trusted. They will cooperatively discard the polynomial after the initialization. It is possible that the mesh router has already been compromised before the group initialization starts. If that happens, our predeployment procedure will fail. To ensure the trustworthiness, the group server can rely on other security mechanisms to ensure the security of initialization process. For instance, Zhang et al. [84] proposed an authentication and key agreement to ensure mutual authentication between clients and mesh routers. Another method is to require

every subgroup leader to get authenticated from the group server in a timely-based order [46].

3.6 Performance Evaluation

We analyze the storage overhead at an end node in HAGK in section 3.6.1, then show the simulation results for rekeying delay and communication cost in section 3.6.2.

3.6.1 Analysis of Storage Overhead

Comparing to a global logical key tree built for the entire group, HAGK has much smaller local key trees. We assume the group size is n and the number of subgroups is g. For a uniform distribution each subgroup will have n/g group members and the height of a local key tree is $logn/g$. Hence each end node only needs to store $(1 + logn/g)$ keys. On the contrary, for a centralized LKH method, $1 + logn$ keys are required for each end node. We plot the storage overhead over the number of subgroups in Fig 3.3 when n is equal to 1024.

We observe that only around half of the keys will be stored for an end node in an medium size WMN with 20 to 50 mesh nodes. As the network size increases, even fewer keys are required. However, in the LKH scheme, the number of keys at an end node is decided by the group size. Therefore, HAGK is more suitable for multicast with a large group over a large network.

3.6.2 Simulation

We use ns2 [5] to perform our experiments. In the simulation, each node represents a mesh router and they remain static throughout the simulations. We simulate the inter arrival time and the average staying time for each member as exponential distribution processes [17]. The parameters we simulated are listed in Table 3.1. For instance, the number of subgroups g ranges from 2 to 100 and the maximum number of each subgroup is set to 1000. The average staying time for each member is set to 30 seconds and the average interval value λ changes

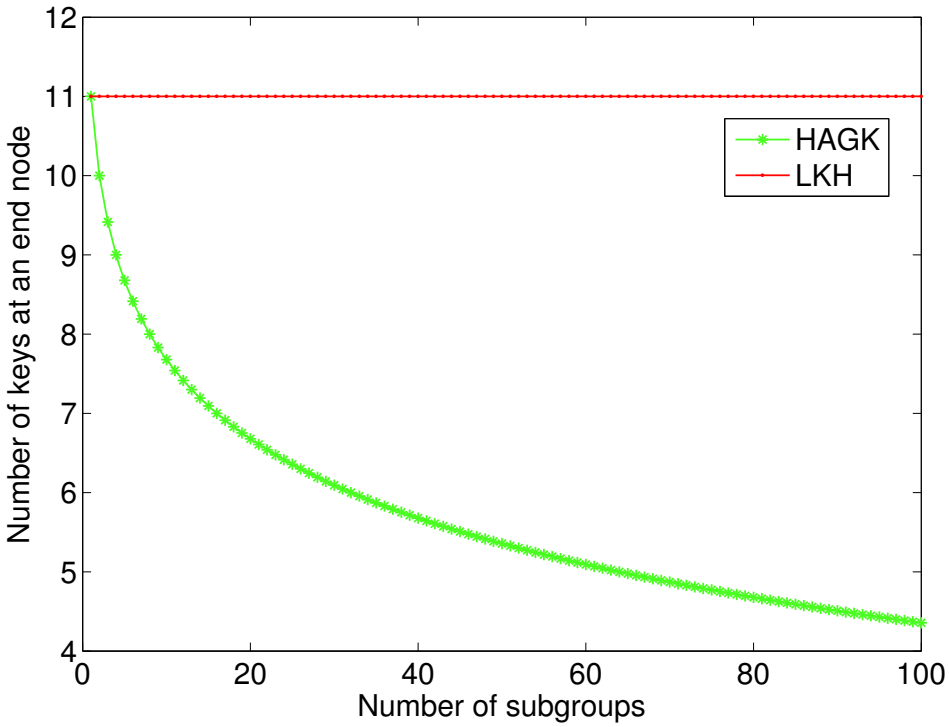


Figure 3.3 Comparison of storage overhead

from 0.1 seconds to 20 seconds. The period for updating the upper layer **session-key** is set to 10 seconds and the simulation time is set to 1000 seconds.

In the experiments, we compared the performance of HAGK with traditional key tree hierarchy in terms of two criteria:

- rekeying delay: Average time required from a key refreshing request is sent to the time when corresponding keys are updated.
- communication cost: average number of messages sent by a node in rekeying when a membership changes.

3.6.2.1 Rekeying Delay

In Fig. 3.4 we plot the average rekeying delay over the network size g. As the results for different network settings are similar, we only show the results when the interval is set to 4.0

Table 3.1 List of simulation parameters of cross-layer based IDS

Parameter	Role
group_size	number of group members
g	number of subgroups
λ	average inter-arrival
γ	average staying time
ρ	period for key refreshment
T	Simulation Time

seconds.

We observe that for both HAGK and traditional LKH group key management the rekeying delay increases as the network size increases. Mathematically, the average delay is very close to a linear function over the network size. However, the increase factor is different. For instance, for a small mesh network with only two or four mesh routers, the rekeying delay for both methods is less than 0.1 seconds. When the network becomes larger (with 100 mesh routers), the average time for key refreshing in HAGK is 0.5 seconds, which is only $\frac{1}{13}$ of the time required for rekeying in LKH group key management.

This is due to the fact that key refreshing in HAGK is localized at the subgroup leaders. Most of key updating only requires communication within one hop when the client is within the transmission range of the associated mesh router. On the contrary, a longer delay is caused in traditional LKH due to the reason that a join request or a leave request may traverse multiple wireless links to reach a group server.

Fig. 3.5 shows how the average rekeying delay is affected by requesting frequencies in different network settings. The average inter arrival time ranges from 0.1s to 20s and the network size n varies from 4 to 100.

Observing the rekeying delay for HAGK shown in 3.5(a), we notice that the average rekeying delay decreases when fewer key updating requests are generated. For example, for a middle size subgroup with 36 group members, the average time to get key refreshed is more than

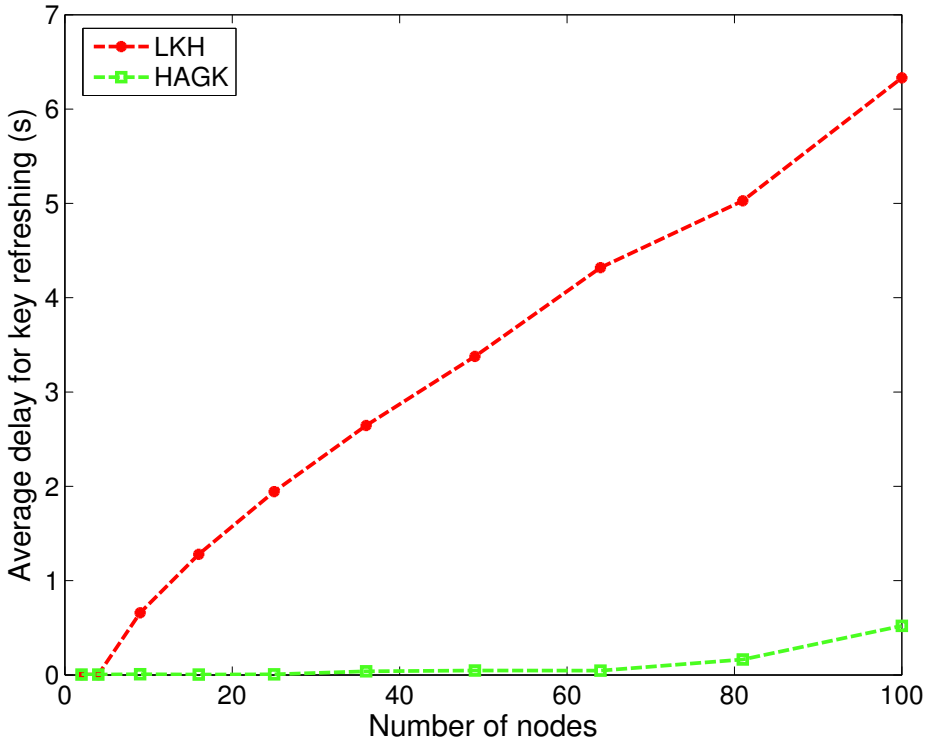


Figure 3.4 Average rekeying delay (*interval = 4.0*)

0.2 seconds when there are average 10 join requests or leave requests per second. If only 5 requests within 100 seconds are generated the delay decreases to 10 miniseconds. This is due to the fact that clients share the medium in the communication with the subgroup leader. Frequent join/leave events eventually cause interferences during the communication. Packets could get dropped and require retransmission. In addition, join/leave requests could get queued up at a subgroup leader.

In our simulation, when a large network is involved, a large group is also created. For instance, when $n = 100$ each subgroup will have maximum 100 members. As a result, the rekeying delay increases as more members are requesting the service at each subgroup.

We observe similar performance for traditional LKH in 3.5(b). Like HAGK, the average rekeying time decreases as inter arrival time increases which generates frequent join/departure events. However, for the same network setting a group member has to wait longer time to join

a group using traditional LKH than join one using HAGK.

3.6.2.2 Communication Cost

We examine communication cost over the backbone node involved in key refreshment for HAGK. Each message could include several keys in our simulation. The average number of messages sent by a mesh node in one second is plotted over *interval* in Fig. 3.6 and Fig. 3.7.

Fig. 3.6 compares the communication cost of HAGK to traditional LKH. For both LKH and HAGK, when *interval* increases, which implies less frequently a rekeying request will be sent, the communication cost decreases. Comparing to LKH, our group key framework incurs more communication overhead within the backbone due to the periodical key updating involved for the top layer key management. However, for a group in which the frequency of incoming rekeying requests is in medium level ($interval = 10$ or $interval = 20$) only two more messages are required for a mesh node.

In Fig. 3.7, we measure the distribution of communication cost in HAGK. The cost involved for the top layer key management and bottom layer key management are plotted. The results show that the cost caused by the periodical upper layer key updating is constant no matter how frequent the membership changes.

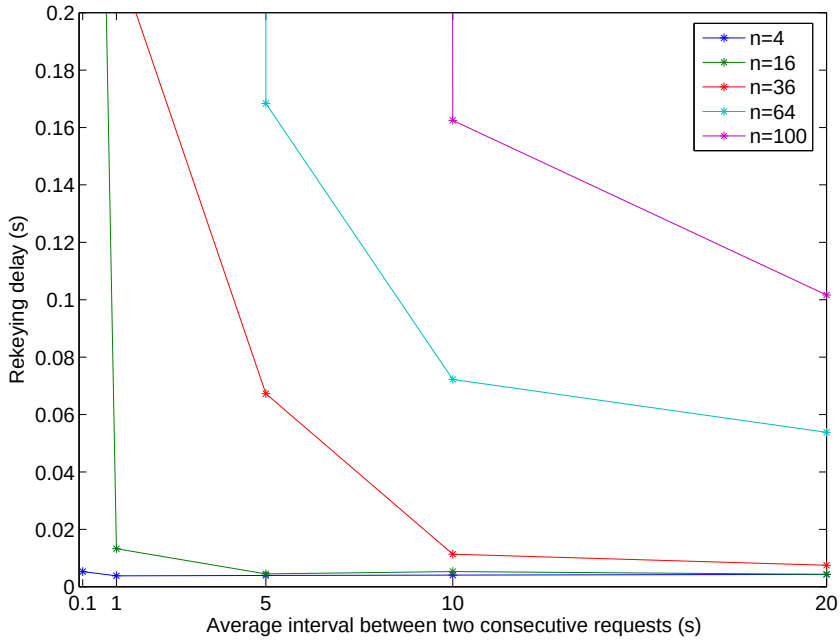
We further measure the communication cost over the number of subgroups in the group key management. For different network topology, the group size remains the same and the number of subgroups is equal to the number of mesh nodes in a WMN. The results for both LKH and HAGK are shown in Fig. 3.8. Notice that the communication cost in HAGK is relatively independent on the size of the network. This is due to the fact that the threshold-based key agreement relies on a localized neighborhood communication and the rekeying requests from member nodes are handled by each subgroup leader. For LKH, as the size of the WMN increases (subgroups increases), the communication cost increases in that key updating messages may broadcast to the whole backbone. The communication cost tends to

be stable after the network size reaches 100 as group size.

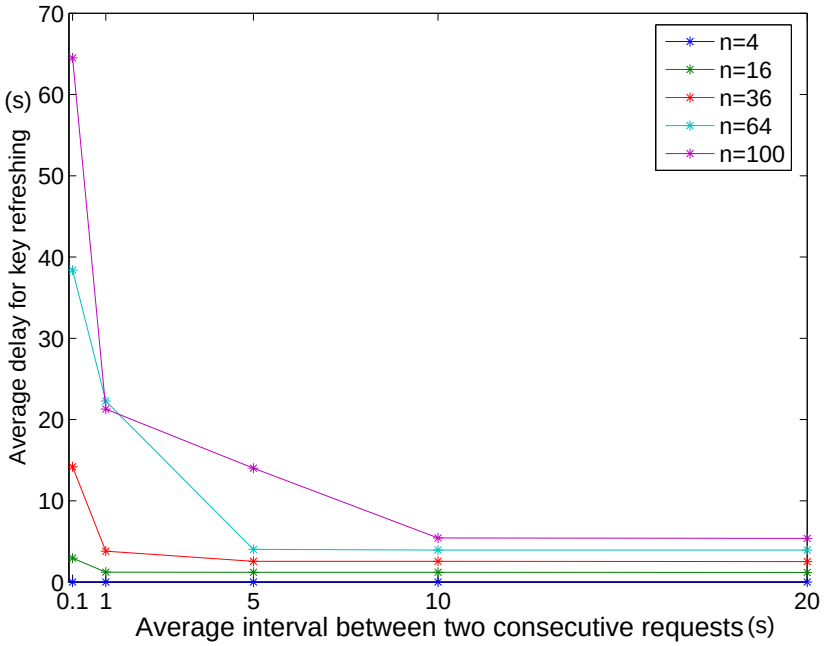
In summary, we observe that our heterogeneity-aware group key management scheme reduces rekeying delay and achieves low storage overhead at end nodes with minimum communication cost within backbone nodes.

3.7 Summary

In this chapter, we proposed a heterogeneity-aware group key management framework which combines the logical key hierarchical technique together with distributed threshold-based techniques in WMNs. We also developed a specific implementation of the group key management framework which addresses the techniques involved in integrating the two different key management schemes. Among them, we provide a Bloom Filter based authentication method and a solution to management member node mobility. The followed analysis and simulation results verify that the proposed method reduces rekeying delay and storage overhead at end nodes with minimum communication cost at backbone nodes. Further, the framework is resilient to both insider and outsider attacks.



(a) Average rekeying delay for HAGK



(b) Average rekeying delay for LKH

Figure 3.5 Rekeying delay over interval

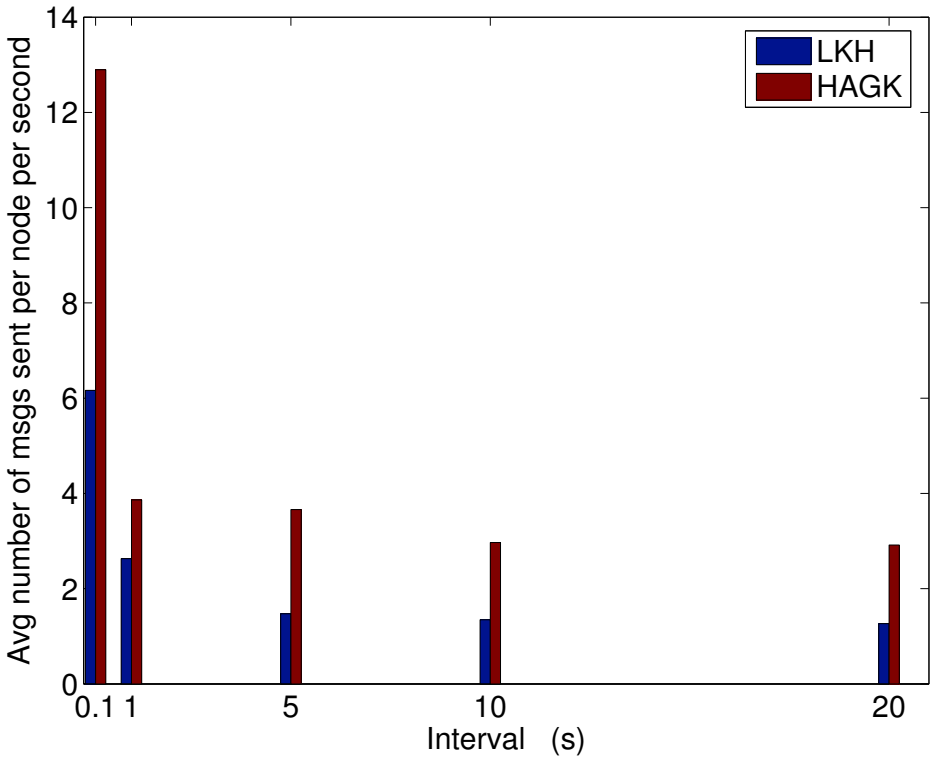


Figure 3.6 Communication cost

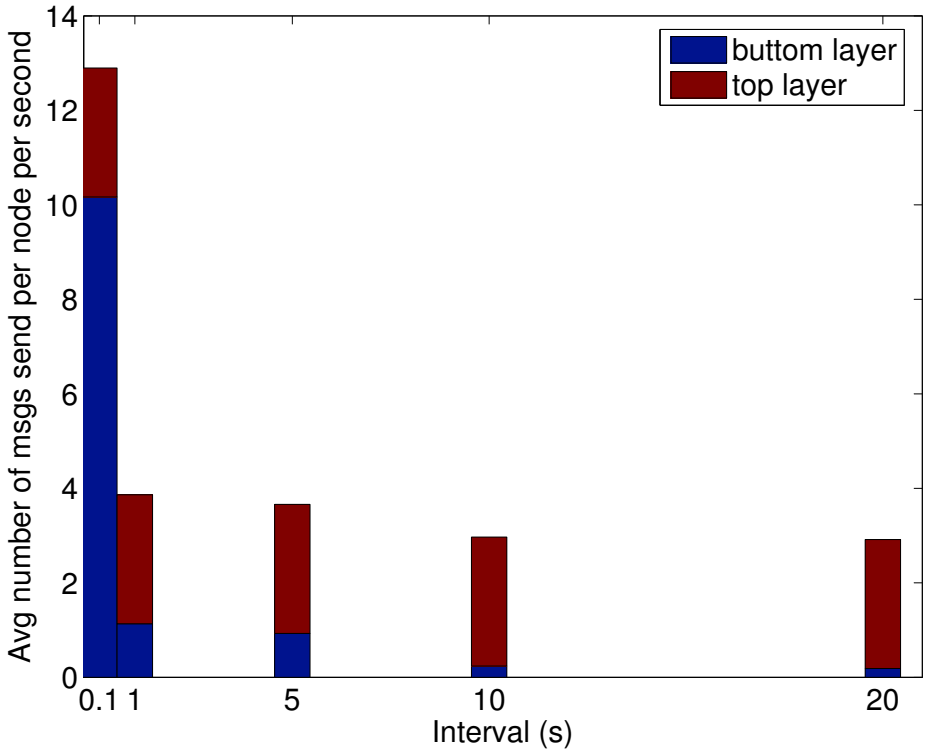


Figure 3.7 Communication cost for session-key update and sub-key update

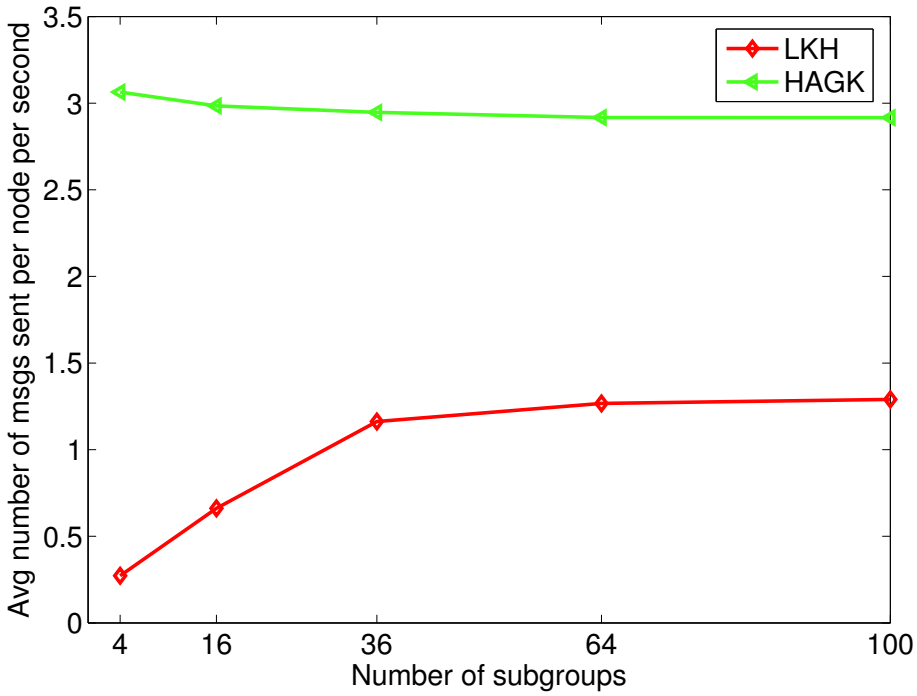


Figure 3.8 Communication for different sizes of groups (the average interval is set to *20* seconds.)

CHAPTER 4. CROSS-LAYER DESIGN OF INTRUSION DETECTION IN WIRELESS MESH NETWORKS

4.1 Motivation

Wireless mesh networking has been a cost-effective technology that provides wide-coverage broadband wireless network services. They benefit both service providers with low cost in network deployment, and end users with ubiquitous access to the Internet from anywhere at anytime. However, as wireless mesh network (WMN) proliferates, security and privacy issues associated with this communication paradigm become more and more evident and thus need to be addressed.

Recently, a few key management schemes have been proposed to ensure secure communication over WMNs. For instance, Zhang et al [84] proposed an attack resilient security architecture for wireless mesh network. Wang et al [76] provides a heterogeneity-aware group key management framework to secure group communication over WMNs. However, utilizing protection and encryption software to protect WMNs are not sufficient and effective, intrusion detection systems need to be deployed to provide a second defense line [85] in such an open environment. We address this problem by designing a cross-layer based anomaly intrusion detection scheme in WMNs. WMN provides more diversified capabilities than mobile ad hoc network (MANET). Its special network architecture and characteristics requires new design in intrusion detection. A WMN is composed of a wireless infrastructure and associated client networks. The wireless infrastructure, normally called wireless mesh backbone, contains a set of mesh nodes (or mesh routers) which could be either static or dynamic and self-form a multi-

hop wireless ad hoc network to relay data from client networks. Various wireless networks, such as wireless ad hoc networks, sensor networks, and WiFi networks, can be integrated to the backbone through the routing and gateway functionalities provided by mesh routers. Due to its open medium, lack of traffic aggregation point, and dynamic topology, WMNs are vulnerable to all attacks that occur in MANETs. However, the mechanisms used in MANETs for intrusion detection is not suitable for WMNs. Mesh nodes are usually equipped with multiple radios. Thus, multiple channels are assigned at each node to support simultaneous data transmission and reception [16, 60]. The integration of various types of wireless networks also requires heterogeneity awareness design in intrusion detection. Further, new challenges are presented in network protocols which integrates link information in routing selection to improve the performance of multi-hop wireless transmission [37].

Many new routing metrics such as Expected Transmission Number (ETX) [34] and Expected Transmission Time (ETT) [37] utilize link quality as the route selection criteria. Due to the high loss rate of wireless links, the new criteria require route selection should be based on link quality instead of solely on hop count. In many cases, the least hop count path may not achieve the best transmission performance. Link quality is normally quantified by bandwidth, loss rate, etc.. In this case, routing protocol is closely correlated with data link layer. In order to accurately catch features when attach happens, cross-layer design can be utilized in intrusion detection over wireless mesh networks. That is, both data link and routing behaviors will be monitored in detecting attacks. Cross layer has long been proposed to optimize the performance and security of network [31] [15]. Network researchers and designers also consider cross layer design for new network protocol stack. Intrusion detection system utilizing cross-layer design are summarized in the related work section of this article.

Currently, research of wireless mesh network is still in its infancy and most of the existing wireless mesh networks are built as testbed or only used for experimental purpose. The lack of practical attacking data and cases requires that our intrusion detection design to be able to accommodate new and novel attacks in WMNs. In this project, we propose to use anomaly

intrusion detection system. Our design goals for such system include the following:

- Increase detection rate by correlating information from multiple layers in the protocol stack;
- Reduce false alarm rate of anomaly intrusion detection using features from MAC layer and network layer;
- Detect cross layer attack targeting different layers in the protocol stack.

Several research effort in cross-layer based intrusion detection design [49, 55, 69, 70] have shown that cross-layer based intrusion detection method outperforms single-layer (e.g. network layer based) based intrusion detection method. However, none of those have been investigated in the WMN environment. In addition, no practical intrusion detection system has been designed and implemented in WMNs.

In this work, we fill the gap by presenting a cross-layer based anomaly detection model which utilizing machine learning algorithms for profile training and intrusion detection. Statistical features from both network layer and data link layer are collected and processed. We have implemented the anomaly detection model on a WMN testbed. Experimental results have shown that cross-layer based method has higher detection rate and lower false alarm rate on the average, comparing to single-layer (e.g. network layer) intrusion detection.

Our contributions lie in the following three aspects:

1. Propose a cross-layer design of anomaly intrusion detection system which exploits the correlation of routing protocols and data link layer in multihop wireless mesh network environment.
2. Design and implement an anomaly based intrusion detection prototype in a wireless mesh network testbed equipped with Microsoft Wireless Mesh Network Toolkit.

3. Evaluate the cross-layer intrusion detection system by comparing its performance with that of a single-layer based intrusion detection system with features only from network layer. Experimental results have validated the effectiveness of cross-layer based anomaly detection methods.

The rest of the chaper is organized as follows. Section 4.2 provides backgrounds for our intrusion detection model. Section 4.3 presents the cross layer detection model and section 4.4 describes the anomaly intrusion detection approach. Section 4.5 introduces the testbed and the intrusion detection software prototype. Section 4.5.3 shows the experimental results. Finally, in section 4.6 we present some conclusions.

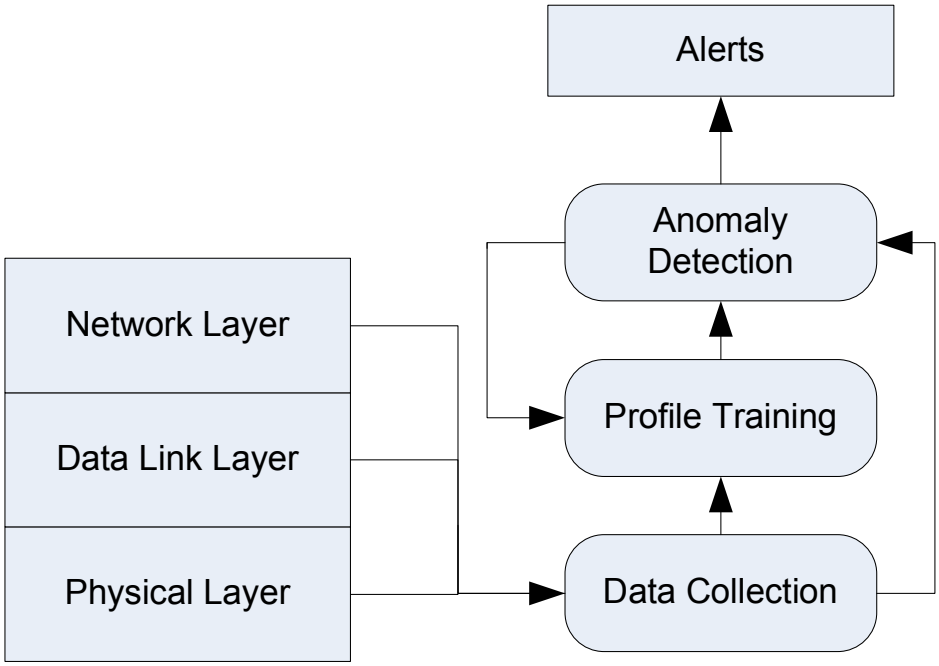


Figure 4.1 Detection model

4.2 Quality Based Routing Metrics

The shortcomings of hop count based routing in multi-hop wireless networks have been recognized in many research efforts ([20, 32, 38–40, 80]). To improve performance of wire-

Table 4.1 Threats in wireless mesh networks

Layer	**Threats**
Physical Layer	jamming, scrambling
MAC Layer	identity theft, rogue mesh node attack, DoS attack, eavesdropping of management messages, management message modification
Network Layer	routing service disruption, traffic pattern distortion

Table 4.2 Feature selection for cross-layer based intrusion detection

Network Layers	**Parameters**		**Statistic Measures**	**Sampling periods**
Physical Layer	Channel switching frequency		Average and Deviation	5 second
Data Link	Bandwidth, Link loss rate and other link metrics		Average and Deviation	
Network Layer	Traffic related	Data packet Route Request Route Reply Route Error	Received, Sent Forwarded, Dropped; Average and Deviation	
	Link cache related	Link added, modified; Route changed	Average and Deviation	

less communication in multi-hop environment, new routing metrics have been proposed. Two typical ones include ETX (Expected Transmission Count) [34] and ETT (expected transmission time) [37]. Both metrics measure the quality of links. The ETX metric measures link quality using expected number of transmissions, including retransmissions, needed to send a unicast packet across a link. ETT measures the expected transmission time of a packet on a link. Further, Drave et al [37] calculate path metric WCETT (Weighted Cumulative Expected Transmission Time) based on ETT by taking into consideration of channel diversity.

MR-LQSR routing protocol is a LQSR protocol with WCETT as routing selection metric. It is a source-based link-state protocol derived from (Dynamic Source Routing) DSR [27] and maintaining properties in link-state routing protocols. For instance, it includes components of neighbor discovery and link information propagation which are similar to corresponding components in link-state routing protocols and modules of route discovery and source route which

it shares with DSR. More details about calculation of ETT, WCETT, and the implementation of MR-LQSR can be found in [37].

4.3 Cross Layer Detection Model

The architecture of our cross layer intrusion detection is illustrated in Figure 4.1. This model includes four components: Data collection module, profile training module, anomaly detection module, and alert generation module. These four modules run on each mesh nodes and collaboratively accomplish the goal of detecting anomaly behaviors in wireless mesh network backbone.

The data collection module gathers audit data from local activities and network traffic according to predefined features sets. In our cross layer design, the data collection module will collect feature data sets from multiple layers which includes channel assignment and channel switching frequency at physical layer, link information at data link layer, and routing activities and data forwarding behavior at network layer. During data training phase, the collected normal data set will be fed into the profile training module for a normal profile generation; during anomaly detection phase, data collection module pipes data set into anomaly detection module for intrusion detection.

Profile training module applies machine learning algorithms to train a normal profile at each mesh node. Three different machine learning algorithms are utilized: Bayesian network, decision tree, and support vector machine (SVM). Those three algorithms are typical machine learning algorithms, and we avoid bias that one single method could bring into the detection.

Anomaly detection module detects intrusion based on the normal profile. Any crossing traffic that deviates from the normal profile is categorized as an intrusion which triggers alert generation and further detection or response. In anomaly detection, any observed behavior that deviates significantly from the profile is considered as an abnormality.

Alert generation module triggers an alarm when an anomaly is identified by the anomaly

detection module. Consequently, further detection may be called to verify the malicious behavior. Such further detection includes but are not limited to global detection, attack detection, and attack source detection. Global detections and attacker tracing back are not addressed in this paper as our focus in this work is to examine the performance of cross layer intrusion detection mechanism and validate its performance by comparing with other single layered design.

4.4 Anomaly Detection

In this section, we discuss the anomaly detection module. Some components in this module is also used in data collection and profile training modules. Anomaly detection is based on the premise that there are intrinsic and observable distinction between normal behaviors and abnormal behaviors. Therefore, features with high information gain, which distinguishes normal behaviors from anomalous behaviors, should be selected. In the following, we analyze threat models in WMNs first, then discuss feature selection based on threat models. The description of classifiers are then followed.

4.4.1 Threats in Wireless Mesh Networks

Wireless mesh network are subject to the same basic threats common for wireless network. In [24], the author analyzed the likelihood, impact and risk of threads at physical layer and MAC layer to the security of the WiMAX/802.16 broadband wireless access technology. Wireless mesh networks suffer from the same type of attacks. We summarize those threats together with attacks [56] at network layer in Table 4.1.

Physical layer attacks are achieved by introducing a source of noise to reduce the capacity of the channel. Those noise can be random (jamming) or purposely targeting specific frames (scrambling). MAC layer suffer from masquerading attacks (identity theft and rogue base station attack), DoS attacks, eavesdropping attack and message modification. In identity theft,

an adversary configures its device to pretend to be some other mesh node. In rogue mesh node attack, a rogue mesh node masquerades a legitimate mesh router to attract clients. At the network layer, two major attacks are involved: one type of intrusions target to disrupt the routing service by manipulating routing information such as sending false routing message or misrouting packets. The other type of attacks breach the normal traffic pattern by dropping data packets or corrupting packet contents.

In WMNs, intrusion at network layer can be classified into routing disruption and data forward disruption. Routing disruption includes all types of attacks that disrupt routing service. Data forward disruption attack includes all malicious behavior that disrupts data forwarding service in WMNs.

4.4.2 Feature Selection

In anomaly detection, we want to select the trace data that bear evidence of normalcy or anomaly. According to [68], MAC layer is to decide who gets access to the medium. A malicious node may always get access to the medium, hence deny the service of other legitimate nodes. As a result, a legitimate node may detect a broken link. At network layer, attackers can target two different disruption: routing disruption and data forwarding disruption. Hence we select those features that can bear those anomalies. In this section, we first define MAC layer features and network layer features, respectively, then present feature sets for each layer.

MAC layer features: statistical values that reflect link state which includes but not limited to loss rate, bandwidth, and link metrics.

Network layer features: statistical values that reflect routing activities. As we specified in section 4.2, LQSR is a link state based dynamic source routing protocol. Every node periodically sends out probe to measure its link forwarding probability and receiving probability which are used to calculate loss probability on the link. If malicious node denies the service on MAC layer, the link will eventually be broken due to high loss rate. Therefore, we select

features on link information which includes link loss rate, bandwidth and the metric that is used to measure link quality.

On network layer, LQSR keeps a link cache which functions as routing table to cache latest and best route to destinations. Also the link cache records each node's link information in the network as every node broadcast their neighboring links periodically. As attacker targeting network layer may disrupt normal routing activities features from routing traffic are logged. For instance, our feature set at network layer includes statistical values of routing request, route reply, routing error, and data packets. Also we select features from link cache. For instance, the statistical value of route changes, link changes.

In summary, Table 4.2 shows the features selected from both MAC layer and network layer.

Different routing protocols may have different features set. In our design, we select LQSR as an example to demonstrate our intrusion detection prototype, which is implemented on the testbed. On the other hand, our feature selection criteria can also by applied to other routing protocols that are using routing quality as the routing metric. For instance, [34] use ETX as routing metric and similar feature sets can be constructed.

Totally, there are 34 features includes in the anomaly detection model, as the average and deviation statistics are used, so there are totally 34 * 2 features in the set. For network-based intrusion detection, only the features in network layer are selected, resulting 20 * 2 features.

4.4.3 Classifier

We use three classifiers in our study. One is a decision-tree classifier, C4.5 (the implementation in Weka [78] is called J48). C4.5 is a typical classifier that computes rules from given feature space to separate data into classes. Another classifier is Bayesian Network (BayesNet algorithm in Weka) which builds probabilistic relationships among features. The encoded Bayesian Network represents the dependencies between features set and classifiers. The last

one is a Support Vector Machine (SVM) classifier, SMO. SMO constructs a separating hyperplane in the given feature space, one which maximizes the margin between the two data sets.

4.5 Experimental Study

4.5.1 Wireless Mesh Network Testbed

The experimental results shown in this paper come from the wireless mesh network testbed, consisting a network of 10 PCs. Each PC is installed with Microsoft Mesh Networking Academic Resource Toolkit [37] and is equipped with two wireless network interface cards. The nodes are placed in offices on two consecutive floors of our computer science department building. The setup is shown in Figure 4.2.

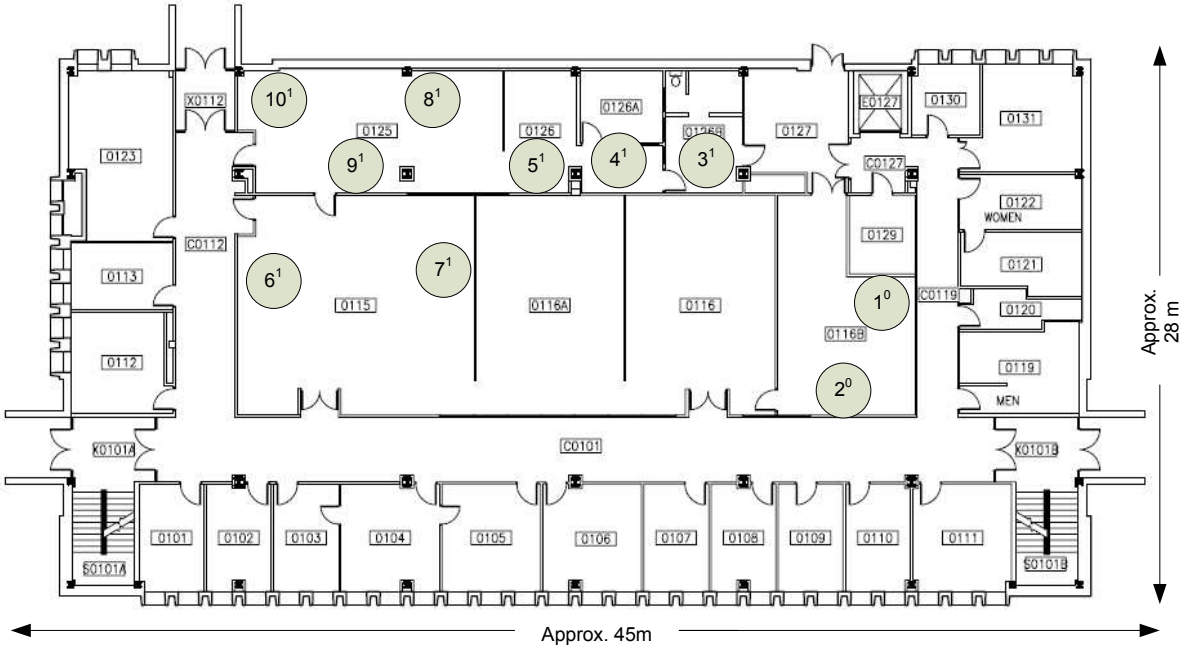


Figure 4.2 A wireless mesh network testbed. Each circle represents a node; the large number is the node ID, and the superscript indicates the floor number the node is on. 0 denotes basement.

The 10 nodes are all Dell PCs. Among them, two have 2.4GHz Intel Core2 CPU with

3GB of memory. Two other PCs have 1.8GHz Intel Pentium 4 CPU with 512MB of memory. The other six nodes have 3.8GHz Intel Pentium 4 CPU with 2.99GB of memory. They all run Microsoft Windows XP. All of our experiments were conducted over IPv4 using statically assigned addresses.

Each node has two 802.11 radios, connected to the PC via PCI slots. Each node has one StarTech 802.11g Wireless PCI NIC Adapter, and also either a 3Com 11a/b/g Wireless PCI Adapter or a Proxim ORiNOCO 11a/b/g PCI Card. Both StarTech NIC adapter and ORiNOCO PCI card have external cable to optimize the placement of antenna. We separate two cards' antennas to one meter to reduce radio interference.

On each node, the two wireless interface cards are both configured in ad-hoc mode with fixed frequency band and channel number. To minimize radio interference [37], we set StarTech cards to use 802.11g with channel 11 and ORiNOCO PCI cards or 3Com PCI adapters to use 802.11a with channel 36. Such a static channel assignment allow each node to have up to 2 links with any of its neighboring node as they could communicate on channel 36 or channel 11.

4.5.2 Implementation of System Prototype

We have implemented our cross-layer based anomaly intrusion detection in the above testbed. Specifically, the data collection module in our anomaly detection is embedded with Microsoft ad-hoc routing framework - Mesh Connectivity Layer (MCL). Our data collection module is incorporated with the implementation of MCL.

MCL is a loadable Windows driver [37]. This driver lies between layer 2 (the link layer) and layer 3 (network layer). It is transparent to upper layer as it appears to be another Ethernet link. To the lower layer, MCl appears to be another protocol running over physical link.

The MCL routes packet using LQSR. The LQSR implementation in MCL is derived from DSR. It includes all the basic DSR functionality, including Route Discovery (Route Request

and Route Reply messages) and Route Maintenance (Route Error messages). The data collection module log all those routing related features from LQSR, which uses a link cache to record all link related information. Therefore, the link related features are collected from the link cache.

In the implementation of LQSR, WCETT is computed based on the link data. Our MAC layer features are all gathered through the WCETT implementation.

Once the raw data is collected through our experiments, they are loaded into pre-process module to generate statistical features. We apply a sliding window scheme to calculate the average and standard deviation of the feature sets. We set a windows size to be k sampling period, the average and standard deviation calculated from the k slots samples will be the final trace data that can be loaded into classifier training module. Then the windows is slided one slot to get the next feature record. When the sliding window reaches the end of the raw data, only the data in the window size is calculated. As the testbed is not synchronized in our implementation, sliding window can be used to avoid experimental bias.

4.5.3 Performance Evaluation

In our experimental study, we compared the performance of cross-layer based anomaly intrusion detection with that of single layer base anomaly intrusion detection. The single layer base anomaly intrusion detection includes only features collected from network layer. Therefore, no link quality features are included in the profile training and intrusion detection.

Two metrics are measured in our comparison:

- *Detection rate*: also called true positive rate which represents the rate attacks are correctly detected. It is calculated as the number of true positives divided by the total number of positives.

- *False alarm rate*: also called false positive rate which denotes the rate normal behaviors are incorrectly identified as intrusions. It is calculated as the number of false alarms

divided by the total number of alarms.

4.5.4 Attack Scenarios in WMNs

To measure the performance of cross layer based anomaly detection, we implemented three different attacks that target at different layers of the network protocol stack: probe flooding attack, greyhole attack, and blackhole attack. We select those three attacks because they are typical attacks that can be launched by an adversary in wireless network and the damage they can cause are severe.

Probe flooding In LQSR, each node periodically sends probe packets to measure its forwarding rate and receiving rate on each of its outgoing links. By doing that, a mesh node can measure the loss probability which is used to calculate the ETT value of that link. As the RTS and CTS are disabled in our experiment, an attacker can launch probe flooding attack to jam the medium. The malicious node gets extra access of the medium by increasing its probe frequency. This attack is a type of DoS attack breaching the fairness at the MAC layer.

Blackhole attack Packet dropping is a very easy but effective attack as an attacker simply drops incoming packets. In our implementation, the attacker drops all packet passing through it which includes routing control packet, data packet, and any packets at MAC layer. This is a cross layer attacks as both MAC layer and network layer are affected.

Greyhole attack Not like blackhole attacker where an adversary drops all incoming packets, greyhole attack only drops certain type of packets, packets from certain users, or packets to certain destination. We implemented greyhole attack as an intrusion at network layer. The attacker only drops routing control packets and data packets at network layer and with a certain threshold (i.e. 10% of the network layer packets).

Table 4.3 Performance of cross-layer based IDS (*n=4*)

Attack Type	J48		BayseNet		SMO	
	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate
Probe Flooding Attack	99.8	0.1	98.6	2.9	99.9	0.0
Grey Hole Attack	98.7	0.3	96.7	0.0	97.7	0.0
Black Hole Attack	98.0	0.4	99.5	0.2	99.3	0.0

Table 4.4 Performance of network layer based IDS (*n=4*)

Attack Type	J48		BayseNet		SMO	
	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate
Probe Flooding Attack	94.3	6.6	90.0	14.8	94.8	7.8
Grey Hole Attack	97.8	0.5	96.2	0.7	96.5	0.0
Black Hole Attack	95.7	1.1	95.1	1.0	98.2	0.3

4.5.5 Experimental Results

In our experiments, two phases are included: profile training and attack detection. Profile training involves all normal network activities. Each node acts as a normal node. The logged data is then used to train a profile model at each node. In attack detection phase, one node in the testbed acts as an attacker running any of the above attacks. Then the data is loaded into Weka for evaluation. Each attack runs for 10% of the running time and each experiment lasts for an hour.

For each attack, we run 5 experiments. For each experiment, the detection rate and false alarm rate at each normal node is calculated. The average results for the 5 experiments represent the performance of each IDS at every node. Then performance of IDSs over the network is measured as the average results from each normal node. The final results are shown in Table 4.3 to Table 4.6. In our experiment, different network size are measured. Table 4.3 and Table

Table 4.5 Performance of cross-layer based IDS (*n=10*)

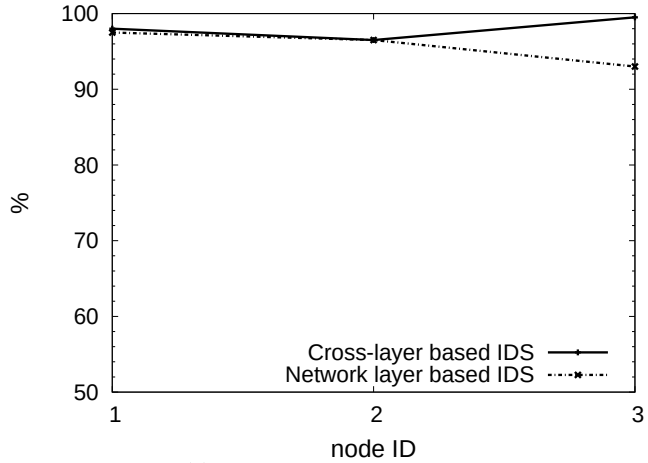
Attack Type	J48		BayseNet		SMO	
	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate
Probe Flooding Attack	97.12	0.32	96.46	0.66	97.63	0.07
Grey Hole Attack	94.76	2.23	89.90	3.88	94.36	2.11
Black Hole Attack	97.5	1.1	95.4	1.2	98.4	0.2

Table 4.6 Performance of network layer based IDS (*n=10*)

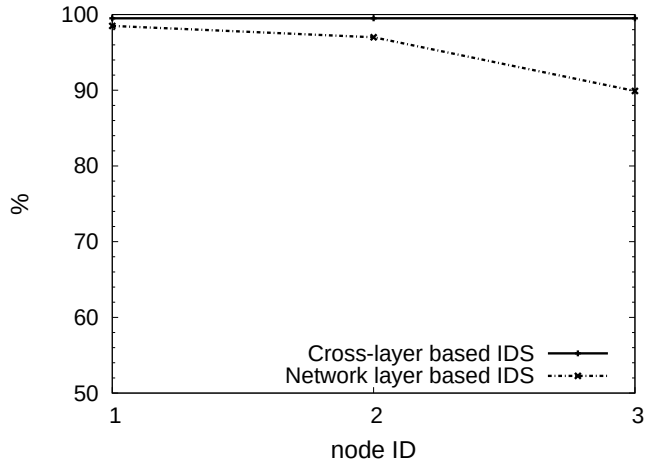
Attack Type	J48		BayseNet		SMO	
	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate	Detection Rate	False Alarm Rate
Probe Flooding Attack	78.81	1.69	76.40	5.51	66.74	1.40
Grey Hole Attack	87.89	3.29	83.31	7.84	84.61	3.67
Black Hole Attack	94.8	1.4	93.4	2.5	95.0	0.5

4.4 display the performance of cross-layer based anomaly detection approach and network layer based IDS, respectively, for a network with 4 nodes and Table 4.5 and Table 4.6 show the results over the testbed with all 10 nodes. The detection rates and false alarm rate for all three attacks using three different machine learning models are shown in the tables.

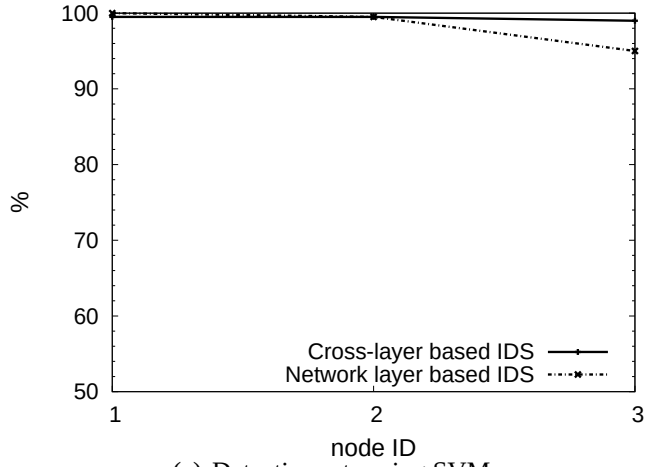
We observe that in both situations, cross-layer based IDS outperforms network layer based IDS. Cross-layer based IDS has higher detection rate using any of the three anomaly models and its false alarm rate is lower than network layer based IDS. For example, with a small network (e.g. with 4 nodes) using BayseNet learning algorithm, the average detection rate for cross-layer based IDS in detecting probe flooding attack is 8.6% higher than network layer based IDS, and its false alarm rate is 11.9% lower than that of network layer based IDS on average. In a larger network (e.g. with 10 nodes), cross-layer based anomaly detection using SVM machine learning algorithm is almost 10% more accurate than network-layer based IDS.



(a) Detection rate using J48

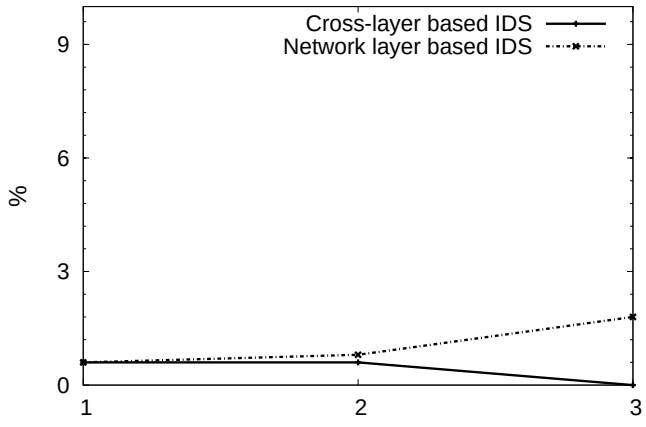


(b) Detection rate using BayseNet

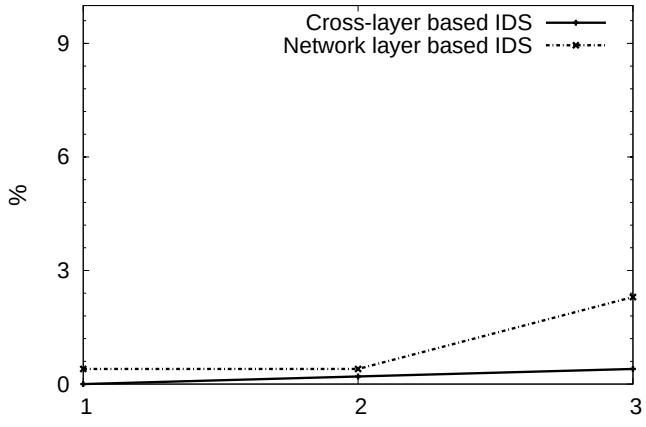


(c) Detection rate using SVM

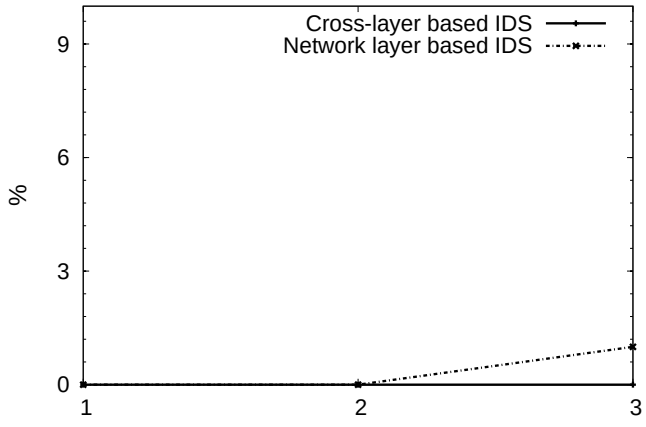
Figure 4.3 Detection rate for blackhole attack (*n=4*)



(a) False alarm rate using J48

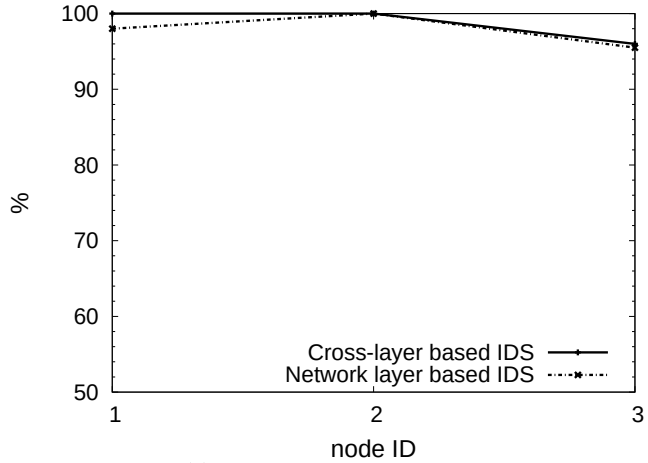


(b) False alarm rate using BayseNet

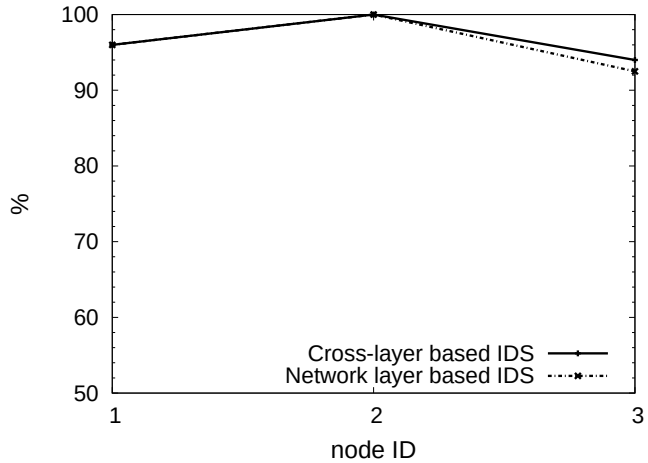


(c) False alarm rate using SVM

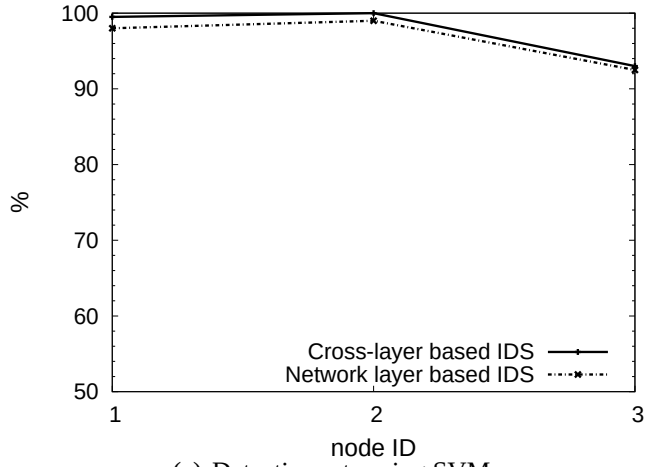
Figure 4.4 False alarm rate for blackhole attack (*n=4*)



(a) Detection rate using J48

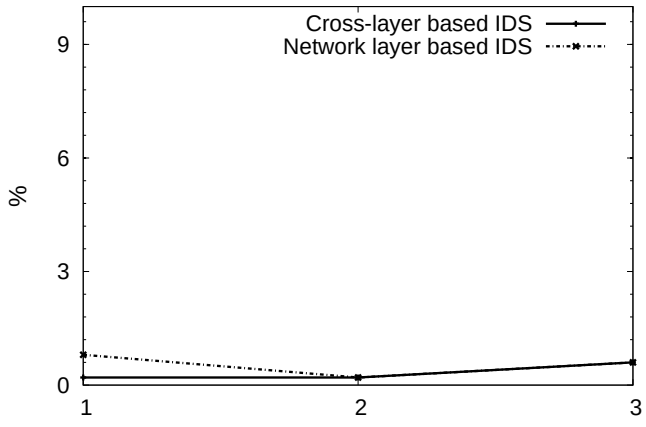


(b) Detection rate using BayseNet

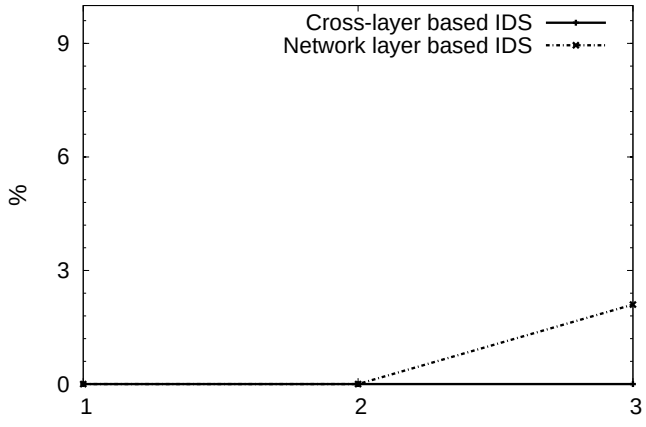


(c) Detection rate using SVM

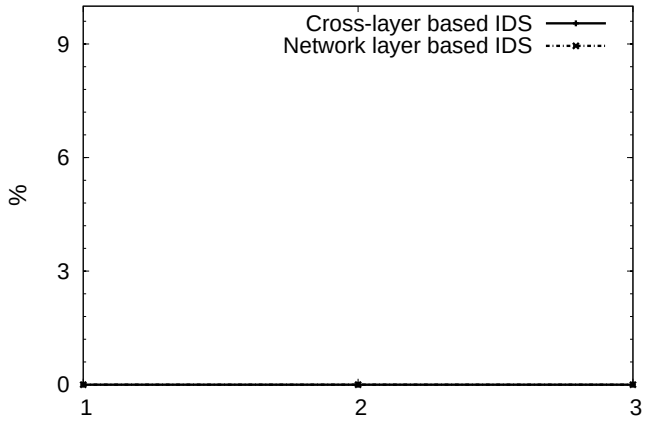
Figure 4.5 Detection rate for greyhole attack (*n=4*)



(a) False alarm rate using J48

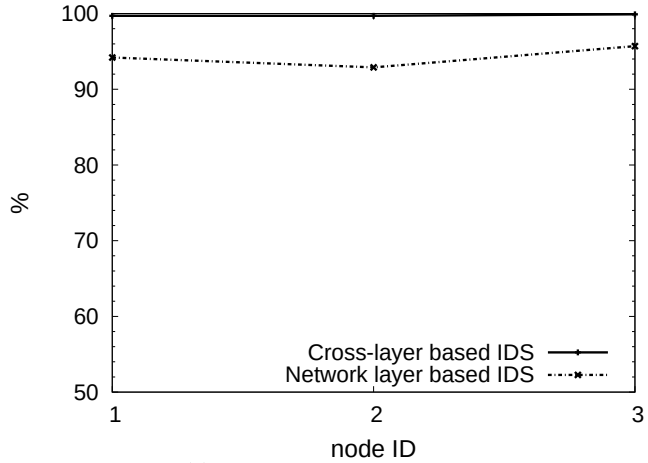


(b) False alarm rate using BayseNet

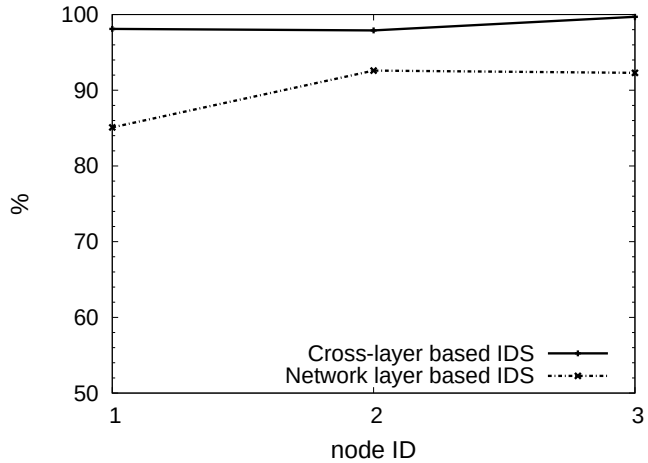


(c) False alarm rate using SVM

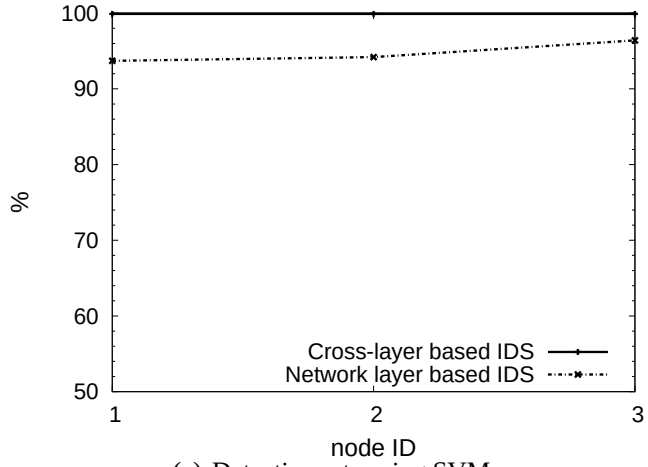
Figure 4.6 False alarm rate for greyhole attack (*n=4*)



(a) Detection rate using J48

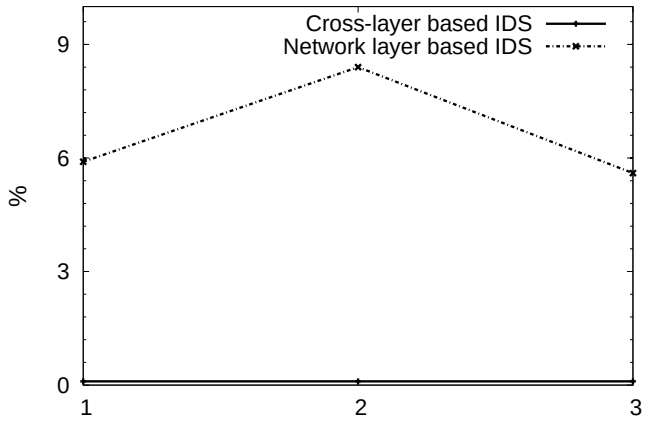


(b) Detection rate using BayseNet

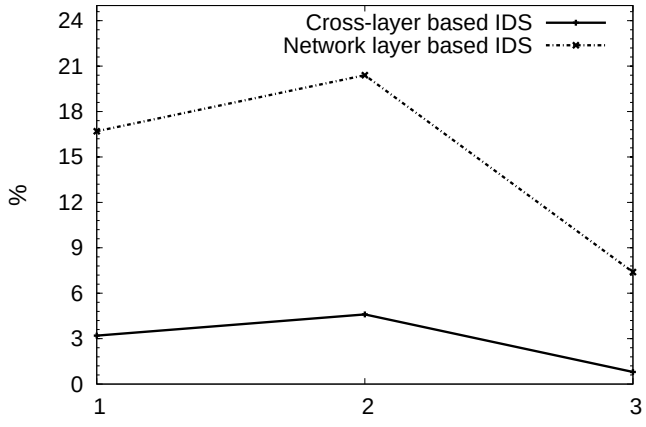


(c) Detection rate using SVM

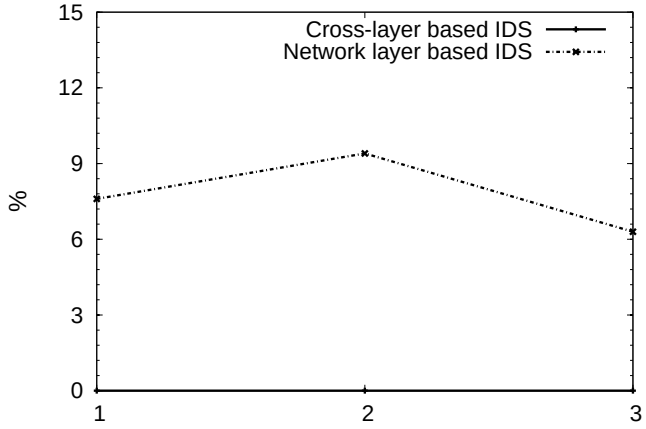
Figure 4.7 Detection rate for probe flooding attack (*n=4*)



(a) False alarm rate using J48

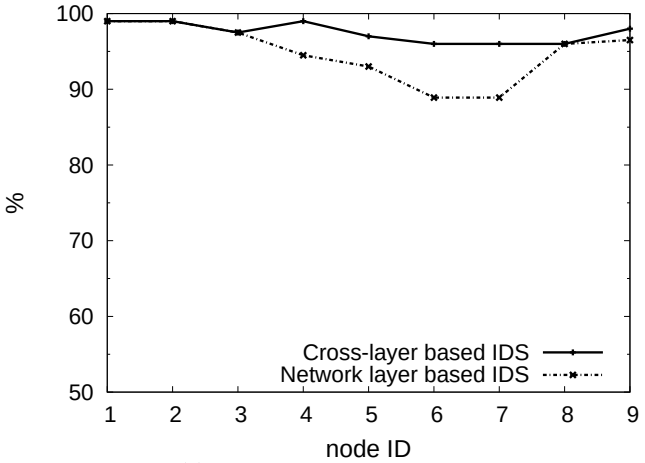


(b) False alarm rate using BayseNet

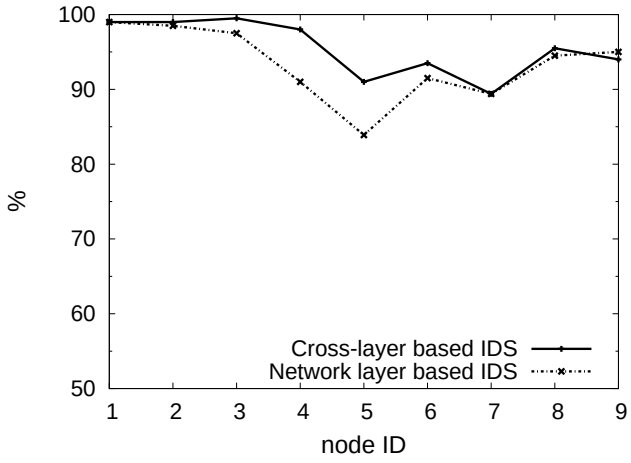


(c) False alarm rate using SVM

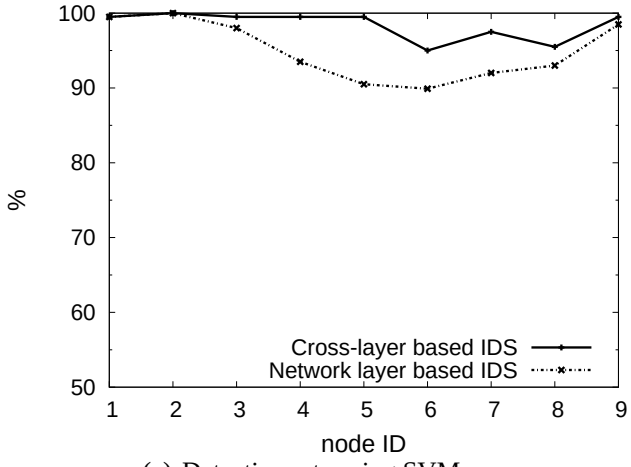
Figure 4.8 False alarm rate for probe flooding attack (*n=4*)



(a) Detection rate using J48

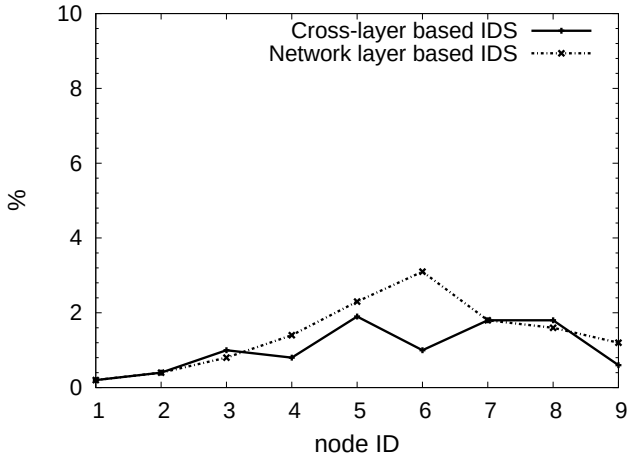


(b) Detection rate using BayseNet

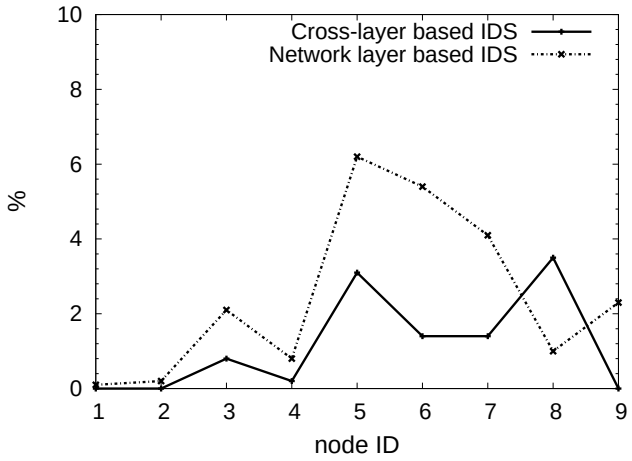


(c) Detection rate using SVM

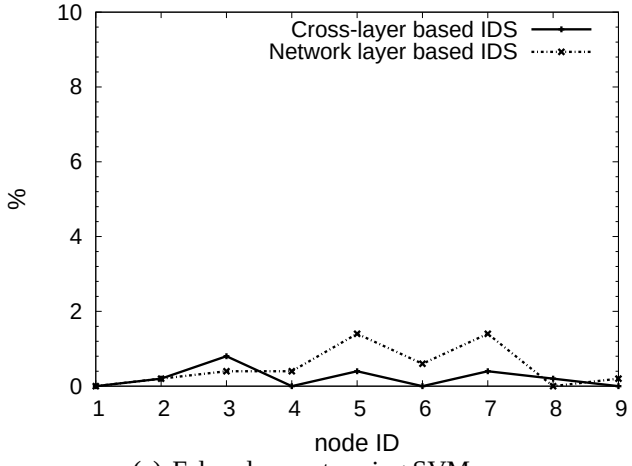
Figure 4.9 Detection rate for blackhole attack (*n=10*)



(a) False alarm rate using J48

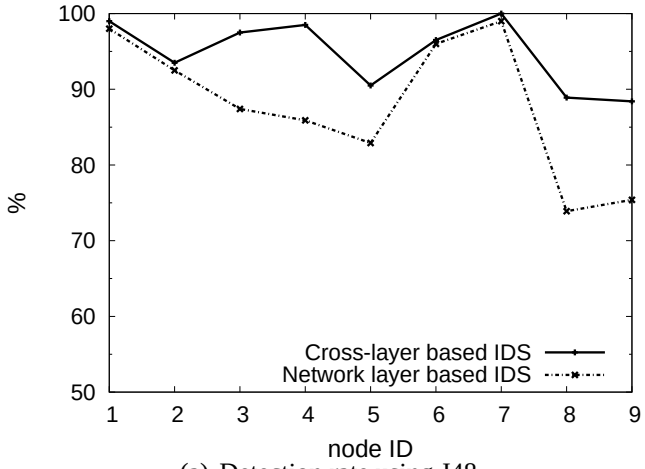


(b) False alarm rate using BayseNet

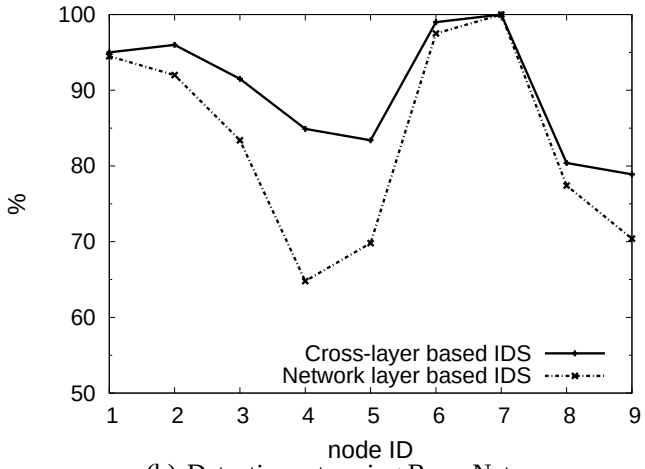


(c) False alarm rate using SVM

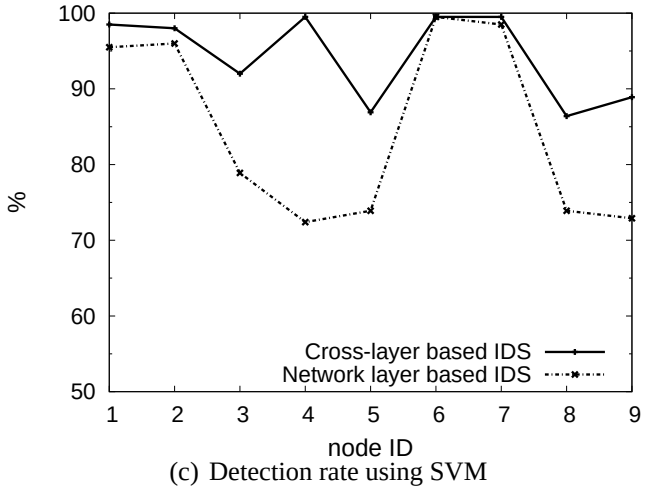
Figure 4.10 False alarm rate for blackhole attack (*n=10*)



(a) Detection rate using J48

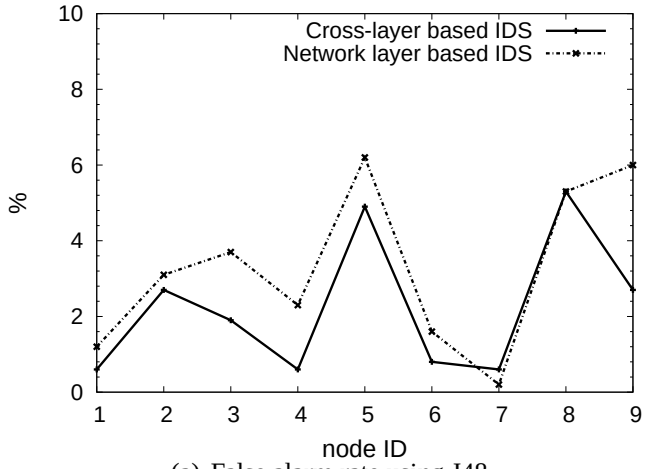


(b) Detection rate using BayseNet

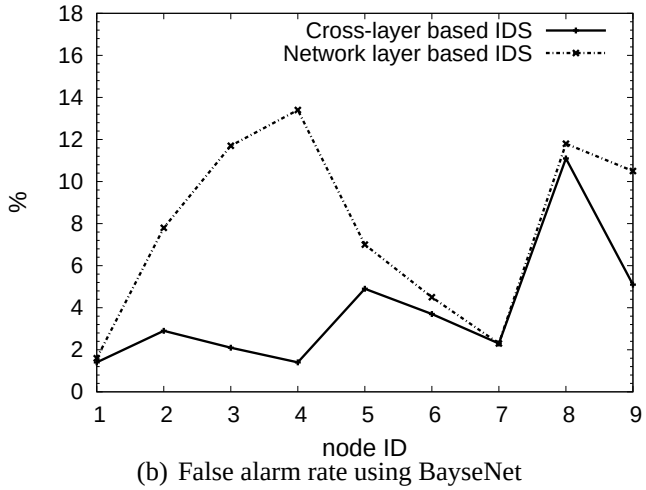


(c) Detection rate using SVM

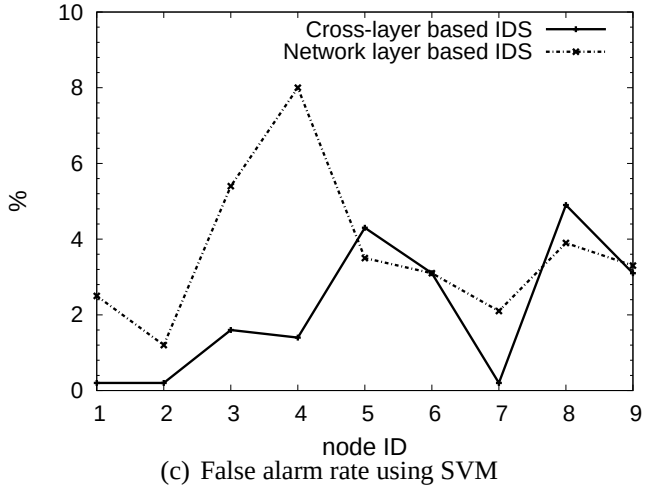
Figure 4.11 Detection rate for grey attack (*n=10*)



(a) False alarm rate using J48

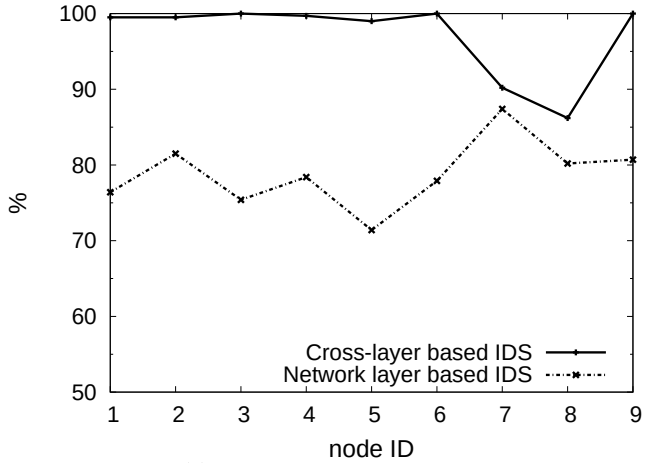


(b) False alarm rate using BayseNet

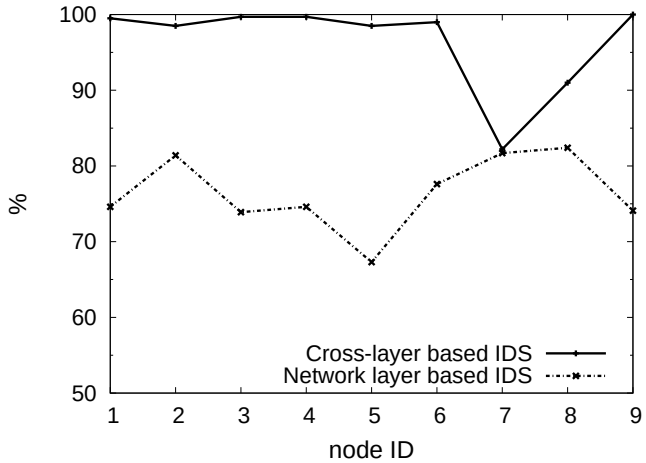


(c) False alarm rate using SVM

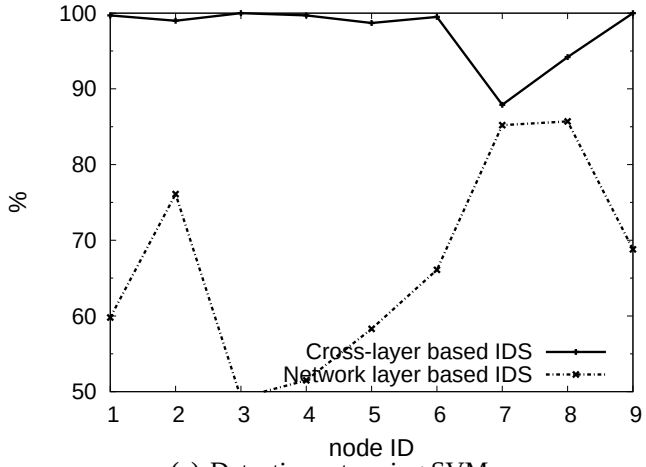
Figure 4.12 False alarm rate for greyhole attack (*n=10*)



(a) Detection rate using J48

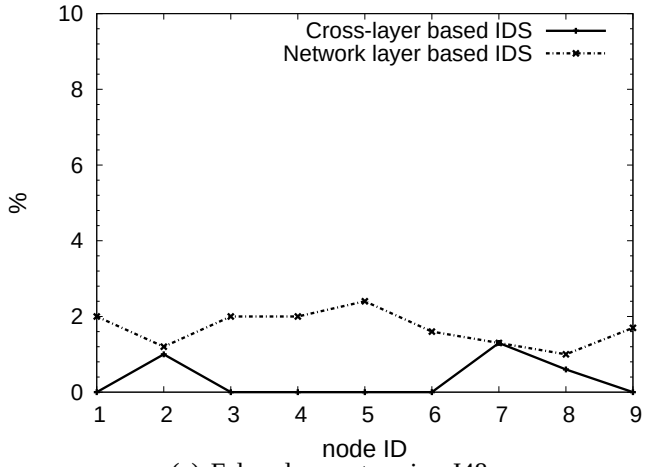


(b) Detection rate using BayseNet

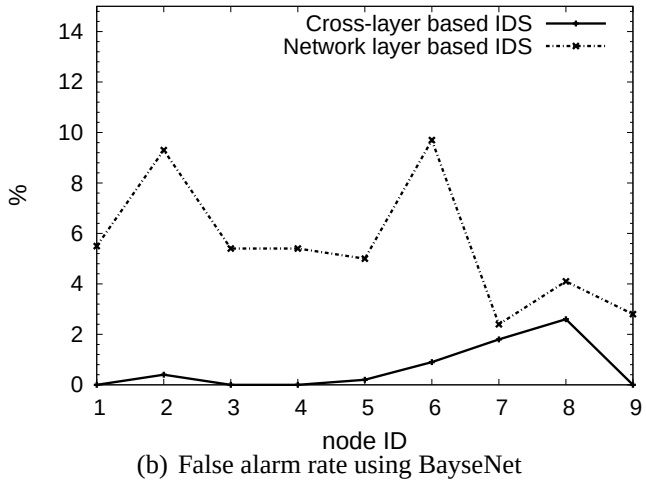


(c) Detection rate using SVM

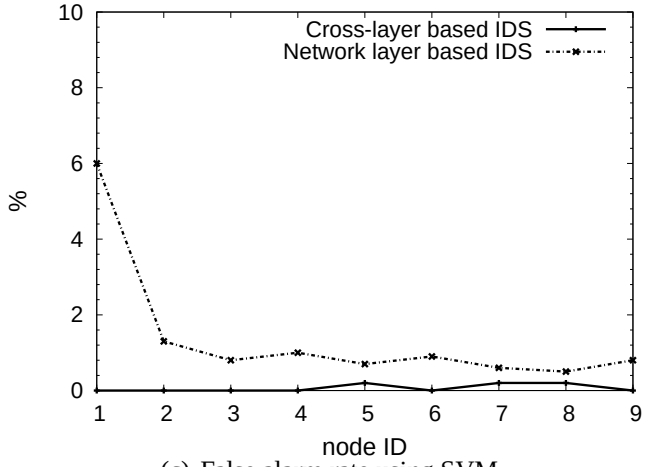
Figure 4.13 Detection rate for probe flooding attack (*n=10*)



(a) False alarm rate using J48



(b) False alarm rate using BayseNet



(c) False alarm rate using SVM

Figure 4.14 False alarm rate for probe flooding attack (*n=10*)

We notice that in detecting MAC layer attack - probe flooding attack - cross-layer based IDS has more significant performance while comparing with network layer based IDS than that in detecting the other two types of attacks. This is due to the fact that cross-layer based IDS also monitors MAC layer activities. Features from both layers are loaded in profile training and intrusion detection.

Another observation is that out of the three learning models, SVM outperforms the other two for both cross-layer based IDS and network layer based IDS. Either it has the highest detection rate or it generates the smallest number of false alarms.

Comparing the performance of both IDSs in the two different size of networks, both IDSs has average higher detection rate and lower false alarm rate in smaller networks. This is due to the reason that smaller networks have short routes in data communication. Therefore, more neighbor nodes can observe the misbehavior of an attack. Over the testbed with only 4 nodes, every node is within one hop distance of the other nodes. The attacker's malicious behavior can be observed by all the other 3 nodes. Therefore, the average detection is more accurate. However, in a large network with 10 nodes, some nodes are 2 or 3 hops away from the source of an attack, the malicious behaviors do not affect those faraway nodes comparing with neighboring nodes, which causes lower detection accuracy in average.

In order to examine the different performance of cross-layer based IDS and network layer based IDS at single node, we plot the detection rates and the corresponding false alarm rates for both IDSs at each normal system. We do not put the results for the attacker as a malicious node can not be trusted. Figure 4.3 to Figure 4.14 show the comparison results. Specifically, Figure 4.3 to Figure 4.8 represents the results for the three different attacks at each of the 3 normal nodes in a 4 node wireless mesh network. And Figure 4.9 to Figure 4.14 show the corresponding results for blackhole attack detection in a 10 node wireless network with one node as the attacker.

For a small WMN (with 4 nodes), the cross-layer based method always outperforms the network layer based IDS at each normal system with higher detection rate and lower false

alarm rate. Especially, for probe flooding attack, cross-layer based IDS is much better than network layer based IDS.

For a large WMN (with 10 nodes), the cross-layer based IDS still holds the trend to have better performance than network layer based IDS. Even though node 3, 8, 9, the cross-layer based IDS does not always has higher detection rate and false alarm rate than network layer based IDS, it doesn't change the fact that the cross-layer based IDS always outperforms network layer based IDS.

4.6 Summary

In this paper, we have presented a cross-layer based anomaly detection system which trains a normal profile from features collected from both MAC layer and network layer. Three machine learning algorithms are used for profile training and intrusion detection. A software prototype of the IDS has been implemented over a wireless mesh network testbed. Experimental studies have shown that cross-layer based IDS outperforms single layer based IDS: on the average it provides higher detection rate and lower false alarm rate.

CHAPTER 5. A GENERIC MODEL FOR INTRUSION RESPONSE

5.1 Introduction

Intrusion detection system (IDS) has attracted a lot of research in the past few decades and has become highly sophisticated, enabling detection of various attacks. Intrusion response, taking action to thwart an attack, is most conducted manually in existing systems. Network administrators have to filter alarms generated by IDS to determine the most appropriate action to take. These manual responses are inflexible and inefficient, especially in distributed systems such as recent wireless mesh network systems. Automated response is required in such systems to respond accurately and quickly.

In most existing intrusion response work, a major concern is how to build the system resource model and use this model to derive the metrics to measure response cost, intrusion cost, and response effectiveness. Many models have been used, including dependency trees [71], graph models [43], and hierarchical tree models [22]. All of these approaches are based on similar foundations which require a system expert or administrator to assign costs to specific resources as well as to build the system model. The cost assignments to system resources, services and users are then used to measure response damage, intrusion damage and response effectiveness. However, directly assigning costs to specific resources is not an easy job for a system administrator who manages the system, and may not be able to accurately estimate the values of the resources in an organization. On the other hand, project managers or company executives may have more accurate estimation for services, but lack the technical knowledge to enumerate system resources. These difficulties have not been addressed in existing auto-

mated response models.

Another drawback of existing models lies in their applicabilities. Most existing models are only applied to estimate response costs, while damage cost evaluations are assumed to be known. In contrast, we propose a generic model to estimate damage cost and response cost to facilitate response action selection. The generic model provides consistency in cost evaluation and response selection.

In this work, we propose to model a system with services and resources, in which services are further divided into application services, component services, and system services and resources are categorized to virtual resources and physical resources. We further present the services and resources in a graph, called dependency graph, in which vertices denote services or resources and edges show the dependencies among them. Each edge in the dependency graph is also marked with appropriate values to describe the dependency weight among services and resources. To estimate the value of each entity (e.g. a server or a resource) the value of the overall system is first estimated by project managers or companies executives. Then the total value of the system is propagated down to other services and resources. After that, the dependency graph with propagated value can be used for damage cost estimation and response cost evaluation according to the affected services and resources.

Our major contributions in this work lie in the following:

- We define a computer system with services and resources and model the system using a dependency graph in which dependencies among services and resources are quantified.
- We describe a top-to-bottom method to propagate the value of a system entity from application services down to physical resources.
- We provide a generic approach in measuring damage cost and response cost.

The rest of the chaper is organized as follows. Section 5.2 provides an overview of response selection criteria. Section 5.3 presents the generic model. Section 5.4 describes the

damage assessment utilizing the model, and section 5.5 shows the response cost estimation. In section 5.6, response selection formulas are discussed. Section 5.7 describes two applications of our model and section 5.8 shows implementation of intrusion response system and experimental results. In section 5.9 we discuss alternative cost propagation function. Other extended usage of the model is also described in this section. Finally, we summarize our work in section 5.10.

5.2 Response Selection Overview

In responding to an intrusion, a system administrator will take several factors into account in selecting a corresponding action:

- *Response Success (RS)*: Response success can also be called as response goodness. It measures how effectively the corresponding response will thwart the intrusion. If a response action will stop the attack completely then it has the full success. That is, the response success should be at least equal to the cost caused by the damage.

- *Operational Cost (OC)*: Deploying a response requires the effort of the system administrator and some other technical support. For instance, the system administrator often needs to spend time selecting a response and launching the action. Cost in generating reports or other related work is also calculated in the operational cost.

- *Response System Impact (RSI)*: A lot of time, a response involves many aspects of the system and may affect other services or resources running on the same system. This effect is counted in response system impact. This is the side effect of an action in responding to an attack.

- *Response Durability* (RD): RD is the expected duration of a response and might not be as important as the other three factors. If two response actions will achieve the same effect in the first three factors, the response that will keep the system longer from

further attacks should be selected. In this case, response durability also reflects how well a response will recover from an attack.

There are also other factors that can be put into the checklist when a response is selected. In existing work, two major methods are used in response selection: One method selects response based on the total cost of deploying a response action. The total cost is calculated by first representing the above factors in a unified unit and then organizing them into a mathematical formula. Another response selection method is based on sorting the importance in accounting those costs. For instance, the system administrator may first select responses with the highest response effectiveness. If more than one response are selected, then the one with the least operation cost may be considered and response durability may come after that.

The two different response selection criteria can be applied to different systems. For example, some system needs to respond to an intrusion using the most effective actions that can minimize the damage of attack and the cost of the response is only considered after that. In this case, priorities are used in choosing the appropriate response action. This normally applies to those systems in which the reputation is more costly. Other system may consider the tradeoff of response effectiveness and the cost paid for deploying the response. The latter one is practical as it may fit certain budget. In our model, the system administrator would have the flexibility to choose either one of them or combines both in reacting to different attacks or in different situation.

5.3 Generic Response Model

In this section, we will first show the description of a system which is divided into resources and application services running above them. Then a generic dependency graph which describes the dependency of services and resources in a system is defined, followed by cost evaluation using this model.

5.3.1 System Elements

A system is a generic term which could encompass a single host, a static network, or a wireless ad-hoc network. It contains applications and resources that support these applications. The applications can be categorized as application services, component services, and system/support services. The set of services is further denoted as S. Services can be any instance of application services AS, component services CS, and system/support service SS. The set of resources are denoted as R. Resources are further divided into virtual resources VR and physical resources PR. The system model is divided into a layered structure.

- *Application Services (AS)*: Application services are those services the system administrator cares about and has an intuitive sense of value for. These might include the web service on a host or DNS service in a standard network. System administrators define relative values for each of the security categories for each of these services.

- *Component Services (CS)*: Component services are those which are user visible, but are not directly used by users. These services typically are subfunctions to support application services. For example, a voice over IP (VoIP) service is composed of VoIP record and replay service, message sending and receiving service, buffer management service, encoding and decoding service, and session management service.

- *System/Support Services (SS)*: System/Support services are services which are required to support component services, but which are not user visible. For instance, the kernel, network service, and file system management service.

- *Resources*: Resources are either *virtual resources (VR)* or *physical resources (PR)* which are required for services to function normally. Physical resources include real hardware such as hard disk, memory, audio card and CPU. Virtual resources contain corresponding driver software or objects stored in those physical resources. Examples are files, sockets, inode, and audio driver. In general, virtual resources would be enough to be

included into a system model. But in many wireless networks such as sensor networks and mesh networks where physical attacks are inevitable, physical resources are also needed in the model for cost evaluation.

For consistency, we call a service or a resource an entity, denoted as e.

5.3.2 Dependency Graph

In an IDS, alerts can result from abnormalities detected at the service level or at the resource level. As responses to detected intrusions, appropriate actions need to be deployed. It is necessary to identify the cost such abnormalities may cause to the system as a whole. In addition, system administrators need to estimate the potential impact a response may bring against the system. The system administrator can determine the cost through values assigned to low-level entities.

However, system administrators may not have an accurate intuition regarding system values to the organization as a whole. In addition, it is difficult to assign cost to low-level services and resources when the system is large. To address these issues we create a graph model to represent a system. This model only requires assigned values at the top-level where estimates can be made more easily. Then the values can be propagated down to low-level entities in the model. We will first give the definition of the dependency graph, then followed with the detail of the propagation function.

Definition *Dependency graph* is defined as a pair (V, E) where V is the set of vertices and E is the set of edges. Any $v \in V$ is a tuple of the form (C, I, A) and we will use $v[C]$ to denote the first element of confidentiality, $v[I]$ to denote the second element of Integrity and $v[A]$ for the last element of Availability in the tuple for the vertex v. We further define $V_C = \{v[C] \mid v \in V\}$, $V_I = \{v[I] \mid v \in V\}$, $V_A = \{v[A] \mid v \in V\}$. Finally, we define the edge relation $E \subseteq Z \times Z$ where $Z = V_C \cup V_I \cup V_A$. To simplify notation, we denote $X = \{C, I, A\}$.

A vertex v in a dependency graph represents an entity (e.g. a service or a resource). $v[C]$

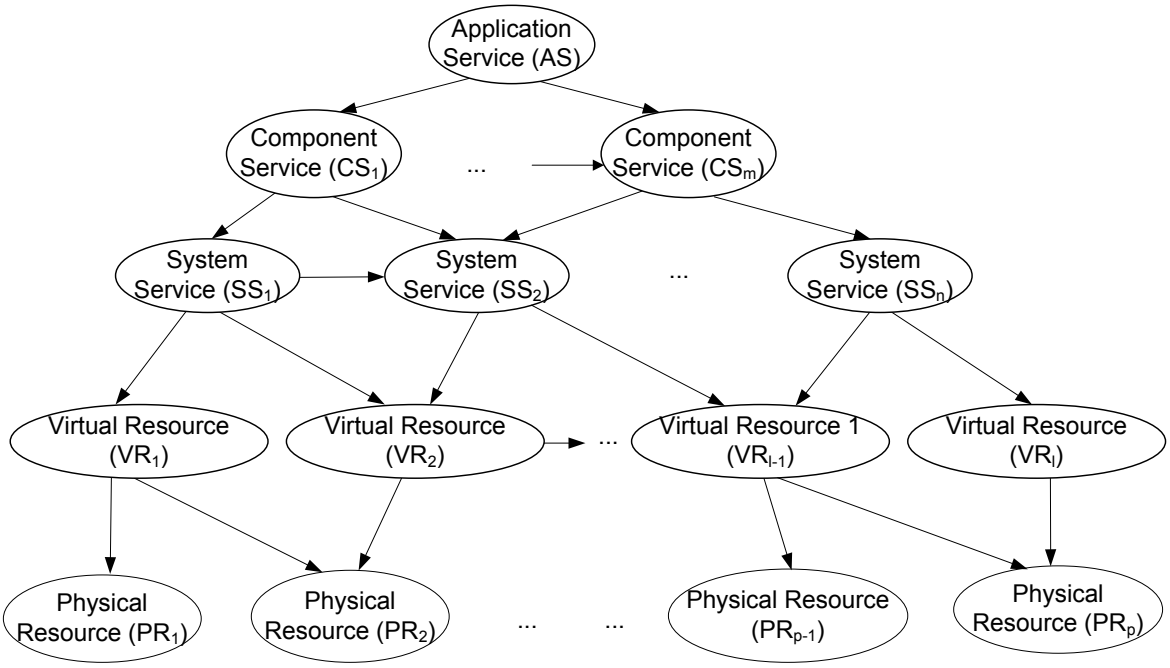


Figure 5.1 A generic system model

denotes the confidentiality of the vertex v, $v[I]$ represents the integrity of the vertex v, and $v[A]$ is the denotation of the availability of the vertex v. An entity v_i is *dependent* on another entity v_j if and only if there exists an edge in E from $v_i[x] \rightarrow v_j[y]$, where $x, y \in X$.

Fig. 5.1 shows an example of dependency graph for a system in which only the dependency relation between entities are presented. For instance, application service AS depends on component service $CS_1, ..., CS_m$, virtual resource VR_{l-1} relies on physical resource PR_{p-1} and PR_p. Fig. 5.2 displays the E relation between C, I, A. For instance, the confidentiality, integrity, and availability of S_i each depends on those of $CS_j, j \in 1, ..., n$.

In Figure 5.3 the dependency graph of a Voice-over-IP (VoIP) system is presented. The VoIP service is divided into eight sub components: recording, replaying, buffering, receiving, sending, session management, encoding, and decoding. Those component services require the support of system services. For instance, all eight component services are dependent on system call; the decoding and encoding services rely on cryptography system; the sending, receiving

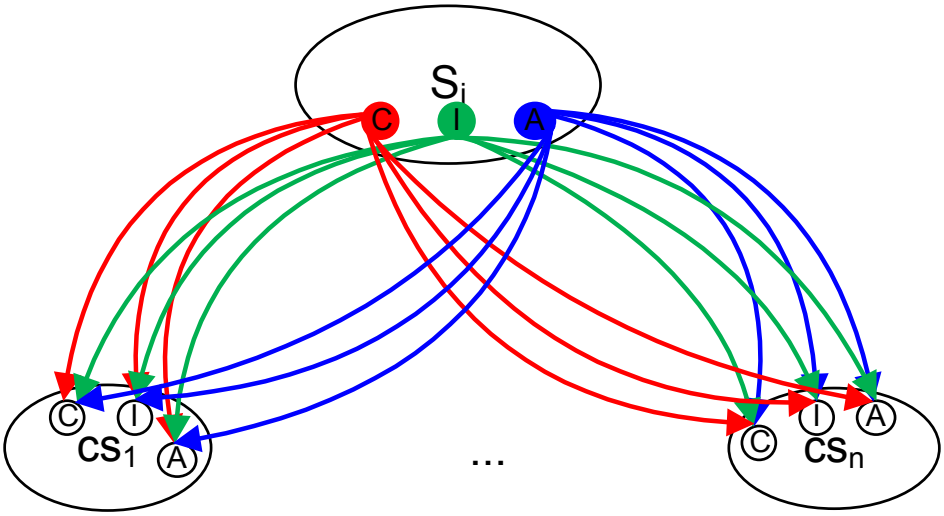


Figure 5.2 An example of weight assignment

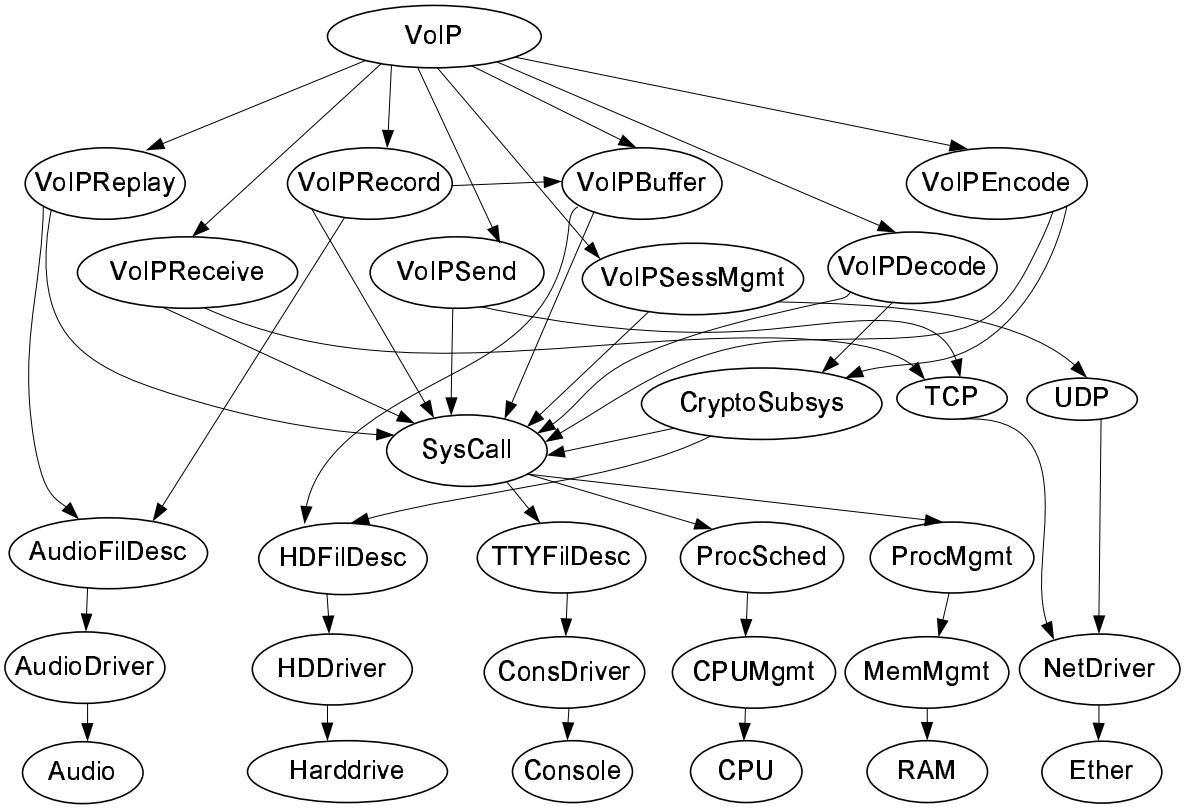


Figure 5.3 Dependency graph of a VoIP application

and session management services are dependent on TCP/UDP service; other system services include process scheduling, process management and file system. All those system services rely on virtual resources: audio driver, hardware driver, console driver, CPU management, memory management, and network driver, whose functionalities depend on the corresponding physical resources.

5.3.3 Cost Propagation Function

In the previous section we defined dependencies among services and resources. However, different systems may have different security goals in terms of confidentiality, integrity, and availability. Some systems may have strict requirements on confidentiality and other systems may demand that availability has the highest priority. For instance, in a VoIP system, confidentiality and availability are highly demanded as the service has to be available to customer and the content of two end customers has to be confidential. In a web service, availability may take the highest proportion of the system value. In addition, the dependencies among properties of services and resources vary in the same system. A good example is the web service. The availability of a web service relies on the confidentiality, integrity, and availability of all its component services. In order to reflect the inter-dependency, a weight is assigned to each dependency edge. For example, the dependency weight between $v_i[x], v_j[y]$ where $x, y \in X$ is denoted as $w_{v_i[x] \rightarrow v_j[y]}$.

In the following, we show a simple approach in assigning dependency weights. In Fig. 5.2, application service S_i is dependent on component services $CS_1, ..., CS_n$. Assume the confidentiality, integrity, and availability of S_i are each equally dependent on the three security properties of CS_j, $j \in \{1, ..., n\}$. In the figure, $S_i[C]$ has $3n$ edges out of the vertex, each points to $CS_j[x], x \in X$. Each edge gets equal weight with the sum equal to 1. Therefore the weight for each edge is calculated as $w_{S_i[C] \rightarrow CS_j[y]} = \frac{1}{3n}, j \in \{1, ..., n\}, y \in X$. Other weights can be calculated similarly.

The weights are subjectively assigned by a system administrator or a system expert as they know more details about the security goals and system configurations. Then the model can be passed to managers or executives for value assignment. The monetary cost of application services can be estimated according to their values to the whole company or institutes.

5.3.4 Cost Evaluation

In this section, we present the cost evaluation methodology for services and resources based on the system model. The monetary value of an application service can be provided by managers/executives as they have a better view of the value of application services to the total economy value or the reputation of the company. Once the high level values are assigned, they are propagated throughout the rest of the system based on the dependency graph.

We use C_v to denote the cost of vertex v, $C_{v[C]}$, $C_{v[I]}$, and $C_{v[A]}$, respectively, to represent the value of each element in vertex v. Before we show the propagation function, we will first give two definitions. For any entity in the dependency graph, we define two costs: *intrinsic cost* C^I and *dependency cost* C^D. The former defines the value of the inner functionalities of entities and the latter expresses the dependency value propagating down to lower level entities

$$C_v = C_v^I + C_v^D$$

We divide the value of an entity into two values as the functionality of the entity is decided by both its inner function and the function of dependent services or resources. When an attack happens at a lower service in the system. However, the intrinsic cost of the upper layer service may not be affected, only the dependency cost which relies on the lower service is affected. To simplify calculation, we assign a percentage to the dependency cost to the total cost, which we denote as P_d, then $C_v^D = C_v \times P_d$, $C_v^D = C_v \times (1 - P_d)$. An application service is a special case as its entire value will be propagated down, thus $P_d = 1$. For all vertex in the lowest layer of the dependency graph, its dependency cost is equal to 0 as there are no more dependent entities below them.

In the following, we show the propagation function for cost evaluation of entities utilizing the dependency model.

1. First, the organization assigns a value to the system (e.g. cost per unit time). This may be derived through direct revenue, loss of productivity if the system is down, or whatever other metric is appropriate. We will call this value C_s with a monetary value per unit time.

2. Next, the organization divides this value between the services provided by the system. If we choose our system to be a server, for example, an organization might assign half of the value to the web service (b), and half to the mail gateway service (m) running on the same server. We represent this by:

$$AS = \{b, m\} \tag{5.1}$$

$$C_b = 0.5 \times C_s \tag{5.2}$$

$$C_m = 0.5 \times C_s \tag{5.3}$$

3. Given this value assignment, the next step is to divide the value between the security goals of C, I, A. In our example, the web service is divided evenly between availability and integrity, and the mail service is divided into half to availability, and a quarter each to integrity and confidentiality:

$$C_{b[C]} = 0 \times C_b \tag{5.4}$$

$$C_{b[I]} = 0.5 \times C_b \tag{5.5}$$

$$C_{b[A]} = 0.5 \times C_b \tag{5.6}$$

$$C_{m[C]} = 0.25 \times C_m \tag{5.7}$$

$$C_{m[I]} = 0.25 \times C_m \tag{5.8}$$

$$C_{m[A]} = 0.5 \times C_m \tag{5.9}$$

4. Once these assignments are made, the system dependency graph is used to propagate these values to the other service and resource sets. A recursive algorithm can be used here. The values of entities at one layer are determined by entities in the same layer and the upper layer. For now we assume that this dependency model is provided, although we will discuss some options for generating and updating such graphs further on.

$$C_{v[x]} = \sum_{u \in V} (\sum_{y \in X} (C^D_{u[y]} \times w_{u[y] \to v[x]})) = \sum_{u \in V} (\sum_{y \in X} ((C_{u[y]} \times P_d) \times w_{u[y] \to v[x]})),\ x \in X. \tag{5.10}$$

Notice that the above algorithm will compute the contribution of each entity to the value of the overall system. As the total values of all applications services are propagated to the first level of component services. Then that value is divided into intrinsic costs and dependency costs. The dependency costs of component services are propagated down to system services, and so forth. Therefore, the total value of all entities that do not fall into application service set is equal to that of all the application service. That is, the following property holds for the dependency graph:

$$\sum_{v \in V \& v \notin AS} C_v = \sum_{u \in AS} C_u$$

5.4 Damage Assessment

Given the dependency graph of a system with the cost evaluated at services and resources, we can use it for cost assessment when a cost of the system is involved. In this section, we show how the intrusion cost can be assessed based on the system model.

For different types of attacks various damage evaluation methods would be applied as the system administrator would have different level of knowledge about the detected intrusion. For some attacks, we may have exact view of its penetration track and the services and resources they affected. The system administrator can map its attack pattern to the corresponding services and resources in the dependency graph. the cost of all affected entities in the dependency

graph will be summed up as the total cost of the intrusion. As the same attack might cause different impact on different systems, its severity will be reflected by the functionality loss to each of its security category. That is, the percentage loss of its confidentiality, integrity, or availability will be measured. For some other attacks that only partial information is available, default actions can be used for cost evaluation. In the following, the algorithm in measuring damage costs for the three different scenarios are presented:

1. *Known attack with signature detected:* A signature-based intrusion detection means the attack pattern can be identified. That is, the affected services and resources are known. When mapping those services and resources to the dependency graph the damage cost can be measured.

   - A set of entities that are affected by the attack is represented as $v_i, i \in 1, ..., m$.
   - The system administrator estimates the damage severity using a function loss rate which represents the reduction of the corresponding functionality. For instance, we use loss rate of $r_i[C]$, $r_i[I]$, and $r_i[A]$, respectively, for the loss of confidentiality, integrity, and availability.
   - The damage cost can be calculated as the sum of all lost intrinsic cost:

$$DC = \sum_{i \in \{1,...,m\}} \sum_{x \in X} (C^I_{v_i[x]} \times r_i[x])$$

2. *Unknown attack with known affected entities* In this case, the signature of the attack is unknown. However, the affected services and resources are detected. We can use the same way for *known attack* to calculate the intrusion damage. That is, the cost of affected entities are added together for the damage cost.

3. *Unknown attack* This is a scenario when new attacks are first detected. The installed intrusion detection system is not able to identify the signature of the attack from its rule storage. And no breach on the services or resources can be monitored. The symptom

of the attack is that some application services might be cracked down. For instance, some services might crash or its loss of availability can be monitored. In this scenario, the system administrator measures the damage cost by estimating the direct loss to the system. For example, if a web site is attacked by unknown attacks, the system administrator may measure the damage cost by estimating the monetary loss of the attack. This can be set to be default in damage estimation. Some parameter can also be set to default, for instance, the loss rate for confidentiality, integrity and availability.

5.5 Response Cost Evaluation

In responding to an intrusion there may be multiple choices in thwarting an intrusion. However, a cost-based model should give higher priority for the response action that causes less cost both in terms of operation cost and negative system impact. In the following, we analyze the three components of response cost: operation cost (OC), response system impact (RSI), and response success (RS).

5.5.1 Operation Cost

Operating cost contains the labor work load involved in deploying the response action. It can be measured as monetary cost in assigning system administrator or technicians in applying the response actions.

5.5.2 Response System Impact

Another component of response cost contains the impact the response brings to the system. Here the system dependency graph can be used. For a response action, the resources and services the response will indirect can be identified. We denote the set as S. Assume the set of vertices affected by the attack is denoted as T. Then the response system impact includes values of all entities affected by the response action, but not by the intrusion. Here we assume

the response will have positive affects on resources damaged by attacks.

$$RSI = \sum_{v_i \in S \& v_i \notin T} \sum_{x \in X} (C^I_{v_i[x]} \times d_{v_i[x]}) \tag{5.11}$$

Where d is the percentage of the response action, which will affect the entity. It takes value in the range of $[0, 1]$.

5.5.3 Response Success

Response success describes how well the response will perform in thwarting the attack. Possible response actions are normally saved in the system in preparation for intrusion response. The entities that the response might invoke can be mapped to the dependency graph. Upon detection of intrusion, corresponding response list will be scanned. The set of entities affected by response actions will be compared with those affected by the attack. Only those entities included in sets will be counted in estimating the value of RS. The calculation is shown as follows.

$$RS = \sum_{v_i \in S \& v_i \in T} \sum_{x \in X} (C^I_{v_i[x]} \times r_{v_i[x]}) \tag{5.12}$$

5.6 Response Selection

In the previous section, we have shown how the dependency graph will be used in estimating damage cost, response cost, and response success. All those parameters are applied to the metric defined in [64, 65] to select a response. Depending on the system requirements specified in section 5.2, different methods could be used in response selection. The basic form of this metric is:

$$EV = RS - (RSI_r + OC_r) \tag{5.13}$$

In the above formula, EV represents *expected value*, which is a measure of value gained by deploying a response in a specific attack context. Values of EV above 0 indicate that the response is worth deploying, and values below 0 indicate that the response will do more harm

than benefit. RS denotes *response success*, or the amount of damage potentially caused by this intrusion which will be mitigated by deploying a specific response. RSI_r is the value for *response system impact* of response r and OC_r is the value *operational cost* of response r. In the calculation, the values of RS, RSI_r, and OC_r are normalized such that the quantity of EV is in the range of $(-1, 1)$.

The first part (RS) of the formula denotes how well the responding action will cover the damage caused by the intrusion. The value of this part would be between 0 and 1. If it is 0, then the response can not cover any of the intrusion damage cost. If it is greater than 0, then the response can cover some part of the intrusion damage.

The second part ($RSI_r + OC_r$) represents the cost the response will bring to the system. It will also be in the range $(0, 1)$, where 0 indicates no cost to deploy, and 1 indicates a cost equal to the entire value of the system.

Therefore, EV give the expected value, relative to the total value of the system, that is associated with a particular response in the context of a specific attack scenario. Full details on the cost computation and components of the function are in [65].

5.7 Scenarios

We provide two different systems to show the application of the generic response model. One system is a wireless mesh network and the other one is a web service system running over a wireless mesh network testbed. In the following, we describe the dependency graph of the two systems and show the attacks and intrusion responses implemented above them. In the next section, we use the wireless mesh network as an example to show the implementation of an intrusion response system and the experimental results.

5.7.1 Wireless Mesh Network System

The mesh network is composed of three nodes. Each node is equipped with two wireless interface cards (NICs). On one NIC, the Optimized Link State Routing Protocol (OLSR) [33] is running for network connectivity. The other NIC is configured for packet sniffing. It monitors network traffic within its transmission range.

We also implement a computer worm to simulate network intrusion. One of the devices in the network is initialized as the source of the computer worm. It selects targets from the routing table and utilizes *Hello* message in OLSR for worm transmission. Once a victim node is attacked, the worm starts to affect other machines in the network from that compromised node. The step will be repeated until no target can be found. More detailed implementation can be found in [50].

5.7.2 Web Service System

In the web service system, an Open Services Gateway Initiative (OSGI) [7] server is installed. Figure 5.4 shows the dependency graph of the web service which is dependent on two other web services - temperature service and speech service - which are exported by OSGI server through its component services - Http service and Axis service. The OSGI service is running on a java virtual machine.

The application is a temperature monitoring service which gets temperature reading from the temperature service and compares it with a threshold. If the value is bigger than the threshold the speech service is called to trigger an alarm.

In our implementation, the dependency weight on the confidentiality, integrity, and availability of services is set to default values:

$$\forall x \in X, u[x] -> v[y], x, y \in X | \exists (u, v) \in E$$

The weight on each dependency link for a single node $(u[x]|u \in V, x \in X)$ is equally assigned. We use $P_d = 0.5$.

5.7.2.1 Attack and Response Implementation

We simulate two attacks in the system:

- *Web Service Attack*: The attacker changes the value exported from the OSGI service to the web service such that alarms will be triggered.
- *DoS Attack*: The attacker reads temperature infinitely which cause the OSGI server (includes OSGI service, Http service, and Axis service) crash.

In Table 5.1, we list the affected services and the corresponding responses for the two intrusions.

Table 5.1 Attacks and intrusion responses

Type of Attack	**Affected Entities**	**Responses**
Web Service Attack	temperature web service	reboot OSGI service
DoS Attack	OSGI service, Http service, Axis service	reboot service, apply access limitation

5.8 Experimental Results

In this section, we are using the wireless mesh network system as an example to show the implementation of intrusion response system and experimental results.

5.8.1 Dependency Graph

In Figure 5.5 the dependency graph of the wireless mesh network is presented. The graph shows the services that the DNS service is dependent on. Unlike a local service which solely based on local services, DNS is a network service which requires services from all hosts in the network. Figure 5.5 reveals such relationship. For instance, the DNS service is dependent on the transport layer service TCP and UDP at each host. Sequentially, TCP and UDP services at

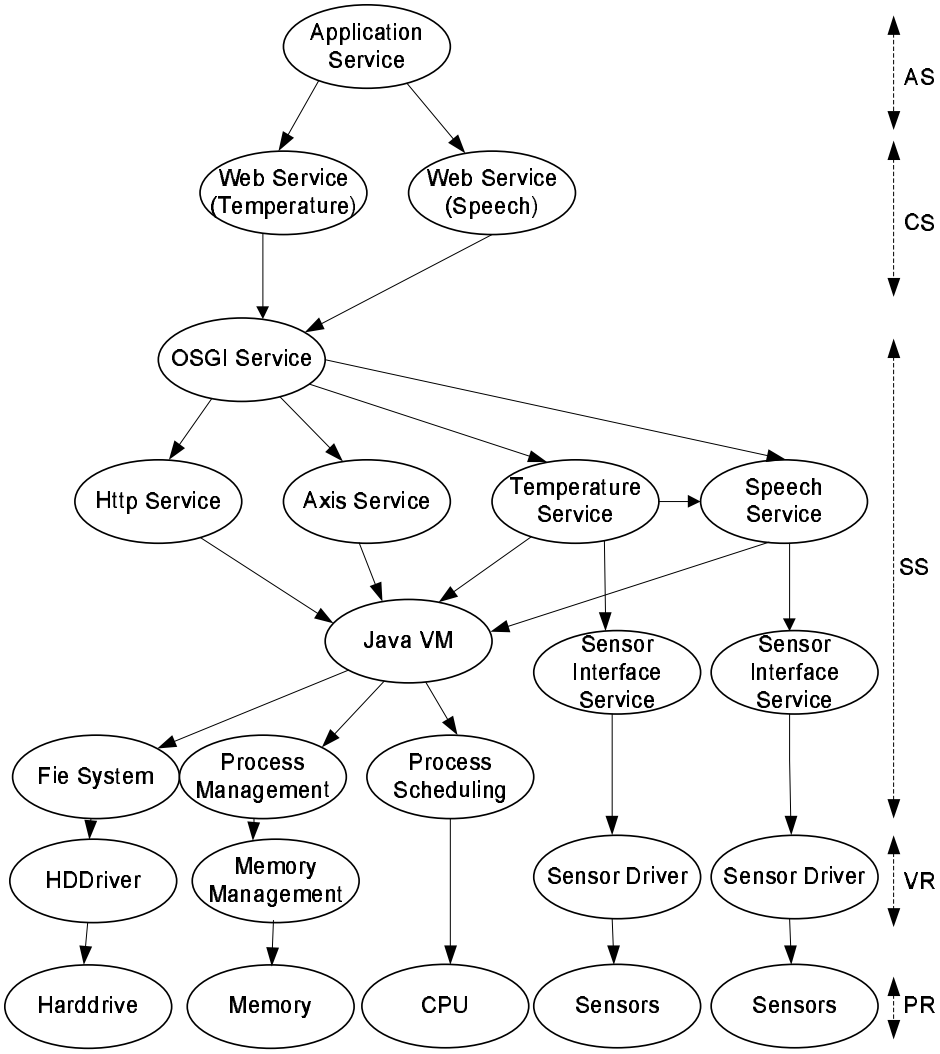


Figure 5.4 Dependency graph of a web service system

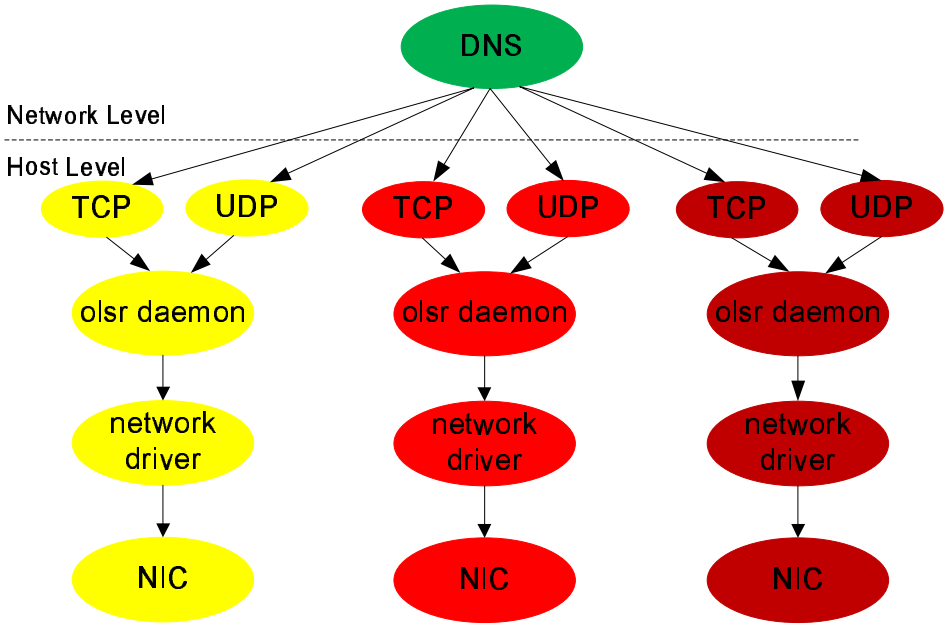


Figure 5.5 Dependency graph of a mesh network with OLSR routing protocol (different colors are used to mark services running on different hosts)

each node are supported by OLSR routing service which relies on the functionality of network driver which controls physical network interface card (NIC).

5.8.2 Implementation of Intrusion Response System

We develop an intrusion response system over the wireless mesh network. Figure 5.6 shows the system architecture. The response system is composed of two modules: intrusion detection module and intrusion response module/engine. The detection module includes different detection sources which are different intrusion detection tools. Using multiple intrusion detection tools can overcome the disadvantage of false alarms inherited in intrusion detection. Once an alert is triggered, the intrusion response engine will be invoked for response selection and deployment. Corresponding information is sent to the response engine through socket.

The intrusion response engine is further divided into initialization module, response selection module, and response deployment module. The initialization module pre-loads the

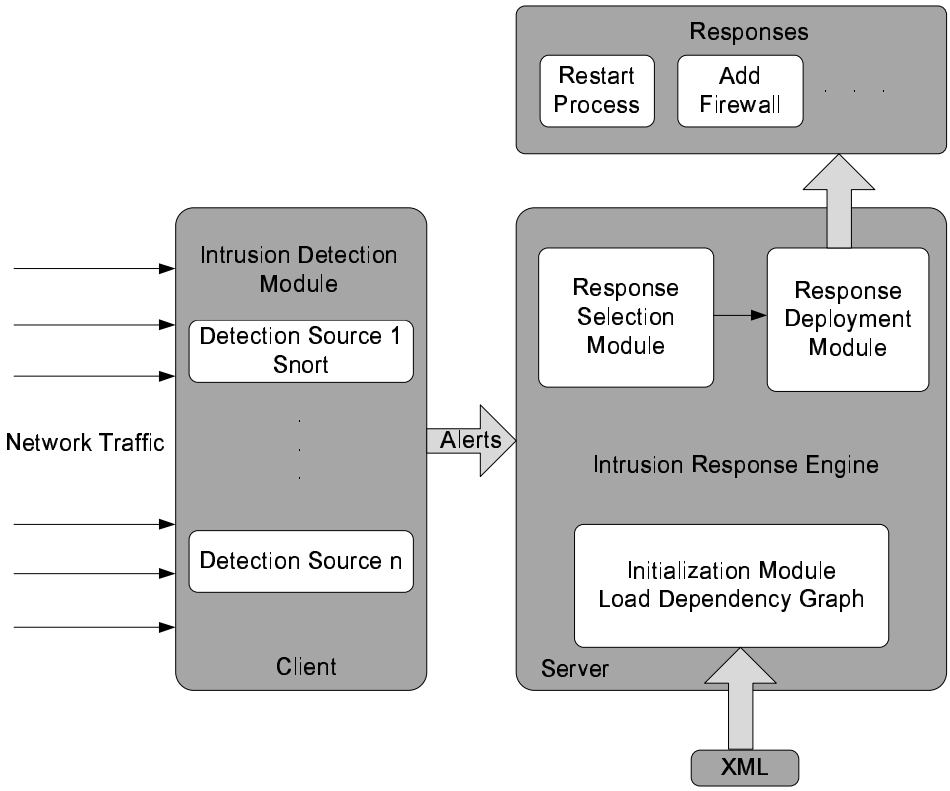


Figure 5.6 Intrusion response engine

engine with the dependency graph which is specified in an XML file. This file also includes a list of response actions and their affecting services and resources in the system. All those information is loaded into the engine during its initialization. For different system, different dependency graphs may specified in the XML, similar to the response implementation.

Once the intrusion response engine starts, it invokes response selection module to choose a proper response action. In this module, factors of operation cost, response system impact, and response success of each response are calculated. Then the expected value EV is computed. The response with the highest EV value which represents the highest gain will be selected. Then the response deployment module will call the corresponding script to start the response.

5.8.3 Implementation of Responses

In response deployment, some responses are general, such as stopping a process, restarting a process, and rebooting the system. Some other responses are specific to certain attacks. In the wireless mesh network, different thwarting actions are implemented based on different detection point of the worm attack. In the following, we first show the attacking process of a worm, then describe the reaction activity and detection results.

Figure 5.7 lists the steps when a worm at *Host* affects another machine *Target*:

1. The *Host* machine sends a crafted Hello message to the *Target* machine
2. The *Target machine* copies the message without checking its size which causes buffer overflow. Eventually, the *Target* machine opens a port for the *Host* machine.
3. The *Host* machine connects to the *Target* machine through the opened port.
4. The *Target* machine sends requests for the worm using Trivial File Transfer Protocol (TFTP).
5. The binary code of the worm will be copied from *Host* to *Target* which will start the above process again.

In the wireless mesh network, Snort can detect different stages of worm attack. Accordingly, three different responses are implemented:

- *Block attacker's IP*: When a worm transmission is detected (step 5) in the neighborhood the third party node will set up firewall rules to block their IPs.
- *Restart OLSR daemon*: When a node is affected its own Snort can detect that and restart the OLSR daemon. We assume that when a machine is affected by a worm its IDS can continue to function.

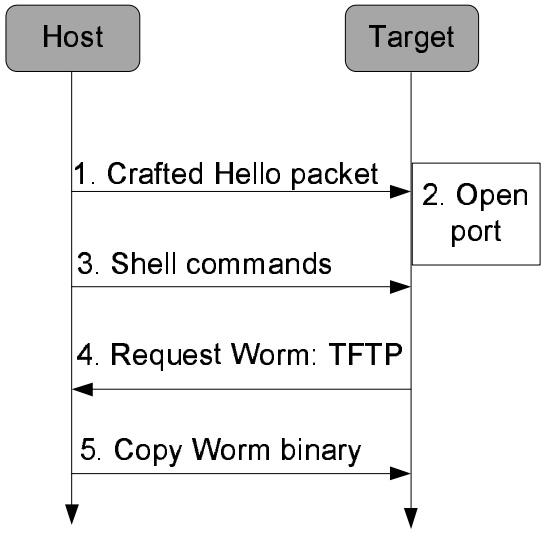


Figure 5.7 Attacking steps for a computer worm

- *Stop TFTP protocol and restart the process*: When a node detects a worm is transmitting to itself, it stops TFTP protocol and restarts the process.

5.8.4 Performance Evaluation

We implement a prototype of our intrusion response system. The dependency graph, attack information and response information are all saved in an XML file and loaded into the system when the intrusion response system starts. We first show the results with intrusion response in the network. Then the performance of the prototype is measured.

Figure 5.8 shows an example of the worm propagating process in the network without deploying intrusion response system. In each step, the worm selects a host as target and tries to copy itself to the victim. For example, in step 1, the worm at *Host2* copies itself to *Host3*. Then it selects *Host1* as the target and affects *Host1* in step 2. In step 3, the worm at *Host3* chooses *Host1* as target. However, *Host1* has been affected in step 2, then it selects *Host2* as the target and affects it. After five steps, all three hosts in the network are affected by the worm and all OLSR daemons die. No network connection is detected in the network.

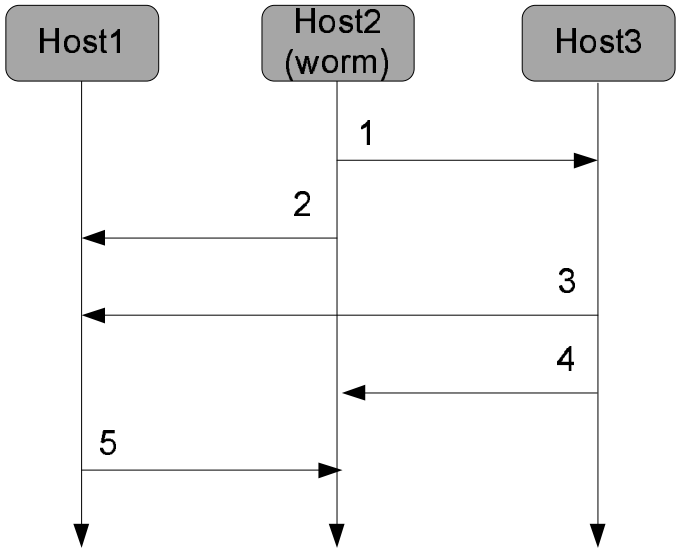


Figure 5.8 An example of worm attack

To compare the results with intrusion response, we install the intrusion response system at each host. The results we observed are as follows:

- In step 1, *Host1* detects a worm transmission within its neighborhood. The response engine at *Host1* selects to *Block attacker's IP*. Then the IP address from *Host2* will be added into its firewall and traffic from *Host2* will be blocked.

- At the same time, the intrusion detection module at *Host3* detects that a worm is transmitting to itself. Therefore, the response *Stop TFTP protocol and restart the process* will be deployed. OLSR daemon will be restarted.

- When *Host2* tries to send a crafted *Hello* message to *Host1* in step 2, it will be blocked by the firewall at *Host1* due to the response in step 1.

- Because of the deployment of intrusion response system, the computer worm failed to affect the network. All three daemons are running and all three nodes are connected by OLSR.

We further evaluate the performance of our intrusion response system. Four metrics will be used for the performance measurement:

- *detection time*: the time from the attack starts till the attack is detected.
- *response preparation time*: the time from the detection of an attack to the alarm reaches the response engine.
- *response selection time*: the time from the response engine is triggered till the response action is selection.
- *response deployment time*: the time from response action is selected till the response is deployed.

Table 5.2 Performance of response selection system

Metric	**Value**
detection time	0.215031 s
response preparation time	4.247133 s
response selection time	0.037682 s
response deployment time	0.036622 s

Table 5.2 shows the performance of one response selection system based on the wireless mesh network. The results are the average value of 10 experiment rounds. Generally, the response system performs well as it has very short time in selection. However, from the results, the response preparation time is much larger than the other three. This is due to our implementation of the connection between intrusion detection system and response system. In the prototype, the response engine periodically scans the intrusion detection for alerts. The period is set to 1 second. When an intrusion is detected, it has to wait for the call from the response system. A more efficient way should allow the intrusion detection system to trigger the response engine whenever it is necessary.

Figure 5.9 shows the results for systems with different number of resources. The resources range from 20 to 10000 which also denotes the size of the dependency graph. As the size of

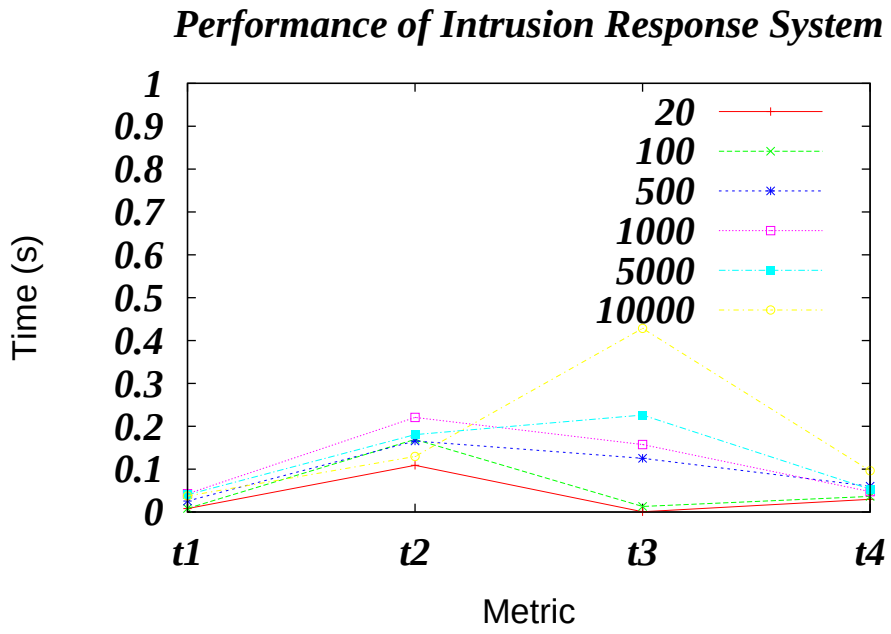


Figure 5.9 Performance of intrusion response system (t1: intrusion detection time, t2: response preparation time, t3: response selection time, t4: response deployment time)

the dependency graph increases, the detection time and response time increase. However, our response system scales very well as it takes less than 0.5 seconds for intrusion detection and response for a large system with 10000 resources.

5.9 Discussion

5.9.1 Alternative Cost Evaluation

In the previous section, the cost of an entity is divided into *intrinsic cost* and *dependency cost*. Only the dependency cost is propagated down to other entities. An alternative method of propagating cost is based on the idea that the functionality of one entity is entirely dependent on other entities. This method can catch the situation when one service is solely dependent on other services. In this case, the cost evaluation method for both response and damage would change as the total value can not exceed that of the application service.

5.9.2 Graph Extension

In addition for response selection, the dependency graph can also be used in other ways.

In response to new attacks, the dependency graph can be used to analyze the possible effect of the attack and identify the scope of the attack to the system. For instance, if the high level service is not affected, the lower level services or resource on which they rely on should not be affected.

Another usage of the dependency graph is to identify key components of the system. Those entities that have the most number of incoming arrows should be guarded by high security level as their failure may cause high cost to the system.

The dependency graph can by extended to provide further details in describing the system. For example, some service component can be further granulated to provide more details of the dependency relationship among entities which can evaluate the response cost and damage cost. At the same time, a large graph can be involved in the calculation and more computation will be invoked. Therefore, it has to be balanced.

5.10 Summary

In this work, we provide a generic model for cost-based intrusion response. A dependency graph is used to represent system resources, services, and the dependency among them. We define the cost propagation function which propagate application value down to component services, system services, and resources. Upon intrusion detection and response, the dependency graph is used for damage assessment and response selection.

To evaluate the intrusion response model, we implement a prototype over two different systems: a wireless mesh network system and a web service system. We show the system dependency graph and measure the performance. Experiments results show that our intrusion response model is effective and scalable.

CHAPTER 6. CONCLUSION AND FUTURE WORK

In this chapter, we summarize our work, review contributions, and discuss the future work.

6.1 Conclusion

WMNs have been a cost-effective technology that provides wide-coverage broadband network services. They benefit both service providers with low cost in network deployment, and end users with ubiquitous access to the Internet from anywhere at anytime. However, as WMNs proliferate, security and privacy issues associated with this communication paradigm become more and more evident and thus should be addressed.

Providing a security system is not an easy job as many aspects have to be considered: Intrusion prevention in which security designs are embedded within the mechanisms; as WMNs are vulnerable to malicious attacks, intrusion prevention are not enough in protecting the network. Thus intrusion detection has to be deployed as a second line of defense to detect attacks in front of unauthorized activities. Once intrusion are monitored and detected, response actions are needed to thwart attacks and minimize the damage caused by the attack. Such a system is referred to as intrusion response system.

In addressing those problems, we provide a systematic framework to protect WMNs. In particular, our research work includes the following major contributions:

1. we proposed a heterogeneity-aware group key management framework which combines the logical key hierarchical technique together with distributed threshold-based techniques in WMNs. We also developed a specific implementation of the group key

management framework which addresses the techniques involved in integrating the two different key management schemes. Among them, we provide a Bloom Filter based authentication method and a solution to management member node mobility. The followed analysis and simulation results verify that the proposed method reduces rekeying delay and storage overhead at end nodes with minimum communication cost at backbone nodes.

2. We have presented a cross-layer based anomaly detection system which trains a normal profile from features collected from both MAC layer and network layer. Three machine learning algorithms are used for profile training and intrusion detection. A software prototype of the IDS has been implemented over a wireless mesh network testbed. Experimental studies have shown that cross-layer based IDS outperforms single layer based IDS: on the average it provides higher detection rate and lower false alarm rate.

3. We provide a generic model for intrusion response. Dependency graph is used to represent system resources, services, and the dependency among them. We define the cost propagation function which propagate application value down to component services, system services, and resources. Upon intrusion detection and response, the dependency graph is used for damage assessment and response selection.

We have conducted extensive simulations to evaluate the performance of our heterogeneity-aware group key management. To evaluate cross-layer based intrusion detection system, we have built a WMN testbed and a system prototype has been implemented. Our intrusion response system is also implemented on the testbed.

6.2 Future Work

Up to now, not many research efforts have been devoted to security issues in WMNs. This thesis provides our initial work in this respect. As a very promising research area, there are

several interesting and important future directions:

- Explore heterogeneity in other security design in WMNs, which integrates various networks requiring new design leverage such as heterogeneity.
- Instead of installing IDS at every node in WMNs, optimal solution can be studied for deployment of distributed IDS in WMNs.
- Extend generic intrusion response model to other system environment. This generic IRS can be applied in other system.
- Integrate the cross-layer based intrusion detection system and intrusion response system such that a complete IDS/IRS can be implemented.

BIBLIOGRAPHY

[1] “Berlin roof net (brn).” [Online]. Available: http://sar.informatik.hu-berlin.de/research/projects/2005-BerlinRoofNet/berlin_roof_net.htm

[2] “Broadband and wireless network (bwn).” [Online]. Available: http://www.ece.gatech.edu/research/labs/bwn/mesh/

[3] “Cisco aironet 1500 series.” [Online]. Available: http://www.cisco.com/en/US/products/ps6548/index.html

[4] “Mesh networking academic resource toolkit.” [Online]. Available: http://research.microsoft.com/netres/software.aspx

[5] “The network simulator - ns2, http://www.isi.edu/nsnam/ns/.”

[6] “Nortel wireless access point.” [Online]. Available: http://products.nortel.com/

[7] “Osgi - the dynamic module system for java.” [Online]. Available: http://www.osgi.org/Main/HomePage

[8] “Purdue university wireless mesh network testbed.” [Online]. Available: https://engineering.purdue.edu/MESH

[9] “Snort, an open source network intrusion prevention and detection system.” [Online]. Available: http://www.snort.org

[10] “Strixsystems.” [Online]. Available: http://www.strixsystems.com/solutions/default.asp

[11] “Tropos networks.” [Online]. Available: http://www.tropos.com/

[12] “UCSB meshnet.” [Online]. Available: http://moment.cs.ucsb.edu/meshnet/

[13] D. Aguayo, J. Bicket, S. Biswas, D. S. J. De Couto, and R. Morris, “MIT roofnet implementation.” [Online]. Available: http://pdos.csail.mit.edu/roofnet/design/

[14] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, “Link-level measurements from an 802.11b mesh network,” *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 121–132, 2004.

[15] I. Akyildiz, X. Wang, and W. Wang, “Wireless mesh networks: A survey,” *Elsevier Computer Networks*, vol. 47, no. 4, pp. 445–487, 2005.

[16] M. Alicherry, R. Bhatia, and L. E. Li, “Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks,” in *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom 2005)*, Cologne, Germany, 2005, pp. 58–72.

[17] K. Almeroth and M. Ammar, “Multicast group behaviour in the internets multicast backbone (mbone),” *IEEE communications Magazine*, 1997.

[18] Y. an Huang, “Intrusion detection and response systems for mobile ad hoc networks,” *Thesis*, 2006.

[19] N. Asokan and P. Ginzboorg, “Key agreement in ad hoc networks,” *Computer Communications*, vol. 23, pp. 1627–1637, 2000.

[20] B. Awerbuch, D. Holmer, and H. Rubens, “High throughput route selection in multi-rate ad hoc wireless networks,” in *Technical Report*, Johns Hopkins University, 2003.

[21] D. Balenson, D. McGrew, and A. Sherman, “Key management for large dynamic groups: One-way function trees and amortized initialization,” *IETF Internet draft*, August 2000.

[22] I. Balepin, S. Maltsev, J. Rowe, and K. Levit, "Using specification-based intrusion detection for automated response," in *the 6th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2003, pp. 136–154.

[23] A. Ballardie, "Scalable multicast key distribution," *RFC 1949*, 1996.

[24] M. Barbeau, "Wimax802.16 threat analysis," in *Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*. New York, NY, USA: ACM, 2005, pp. 8–15.

[25] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, 1970.

[26] B. Briscoe, "MARKS: Multicast key management using arbitrarily revealed key sequences," in *Proceedings of the First International Workshop on Networked Group Communication*, November 1999.

[27] J. Broch, D. Johnson, and D. Maltz, "The dynamic source routing protocol for mobile ad hoc networks," *http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt*, Dec. 1998.

[28] S. Buchegger and J.-Y. le Boudex, "Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks," in *the 10th Euromicor Workship on parallel, Distributed and Network-based Proceeding*. IEEE Computer Society, 2002.

[29] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *INFOCOMM'99*, 1999. [Online]. Available: citeseer.ist.psu.edu/canetti99multicast.html

[30] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," *Lecture Notes in Computer Science*, vol. 1592, pp. 459–474, 1999. [Online]. Available: citeseer.ist.psu.edu/216853.html

[31] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," in *Proceedings of IEEE*, January 2007.

[32] K.-W. Chin, J. Judge, A. Williams, and R. Kermode, "Implementation experience with MANET routing protocols," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 5, pp. 49–59, 2002.

[33] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," in *RFC 3626*, Oct. 2003.

[34] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003, pp. 134–146.

[35] B. DeCleene1, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure group communications for wireless networks," in *Proceedings of the MILCOM*, June 2001.

[36] L. R. Dondeti, S. Mukherjee, and A. Samal, "A dual encryption protocol for scalable secure multicasting," in *In Fouth IEEE Symposium on Computers and Communications, Red Sea, Egypt*, 1999, pp. 2–8.

[37] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceeding of ACM MobiCom*, September 2004.

[38] R. Dube, C. D. Rais, K. yeh Wang, and S. K. Tripathi, "Signal stability based adaptive routing (ssa) for ad hoc mobile networks," *IEEE Personal Communications*, vol. 4, pp. 36–45, 1997.

[39] T. Goff, N. B. Abu-ghazaleh, D. S. Phatak, and R. Kahvecioglu, “Preemptive routing in ad hoc networks,” in *In Proc. ACM/IEEE MobiCom*, 2001, pp. 43–52.

[40] Y.-C. Hu and D. Johnson, “Design and demonstration of live audio and video over multihop wireless ad hoc networks,” in *MILCOM*, 2002.

[41] Y.-C. Hu, A. Perrig, and D. B. Johnson, “Rushing attacks and defense in wireless ad hoc network routing protocols,” in *Technical Repor TR01384*, Depamnent of Computer Science, Rice University, 2002.

[42] D. Huang and D. Medhi, “A key-chain-based keying scheme for many-to-many secure group communication,” *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 4, pp. 523–552, 2004.

[43] M. Jahnke, C. Thul, and P. Martini, “Graph based metrics for intrusion response measures in computer networks,” in *32nd IEEE Conference on Local Computer Networks (LCN)*, Dublin, Ireland, October 2007.

[44] D. B. Johnson, D. A. Maltz, and Y. Hu, “The dynamic source routing protocol for mobile ad hoc networks (DSR),” *IETF MANET Internet Draft*, 2003.

[45] Y. Kim, A. Perrig, and G. Tsudik, “Simple and fault-tolerant key agreement for dynamic collaborative groups,” in *ACM Conference on Computer and Communications Security*, 2000, pp. 235–244.

[46] J. T. Kohl and B. C. Neuman, “The kerberos network authentication service v5,” in *RFC4120*, 2005.

[47] W. Lee, W. Fan, M. Miller, and S. Stolfo, “Toward cost-sensitive modeling for intrusion detection and response,” *Journal of Computer Security*, vol. 10, pp. 5–22, 2002.

[48] L. Liao and M. Manulis, “Tree-based group key agreement framework for mobile ad-hoc networks,” *Future Generation Computer Systems*, vol. 23, no. 6, pp. 787–803, 2007.

[49] Y. Liu, Y. Li, and H. Man, “A distributed cross-layer intrusion detection system for ad hoc networks,” in *First International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm05)*, Athens, Greece, September 2005.

[50] J. R. Lynch, “Intrusion detection systems in wireless ad-hoc networks : detecting worm attacks,” *Thesis (M.S.)*, 2006.

[51] G. T. M. Steiner and M. Waidner, “Key agreement in dynamic peer groups,” *IEEE Transactions on Parallel and Distributed Systems 11*, vol. 8, pp. 769–779, 2000.

[52] D. A. Maltz, J. Broch, and D. B. Johnson, “Lessons from a full-scale multihop wireless ad hoc network tesbed,” *IEEE Personal Communications*, pp. 8–15, February 2001.

[53] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” in *ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[54] S. Mittra, “Iolus: A framework for scalable secure multicasting,” in *SIGCOMM*, 1997, pp. 277–288. [Online]. Available: citeseer.ist.psu.edu/mittra97iolus.html

[55] J. Parker, A. Patwardhan, and A. Joshi, “Cross-layer analysis for detecting wireless misbehaviour,” in *IEEE Consumer Communications and Networking Conference Special Sessions (CCNC)*, Las Vegas, Nevada, January 2006.

[56] A. Patwardhan, J. Parker, M. Iorga, A. Joshi, T. Karygiannis, and Y. Yesha, “Threshold-based intrusion detection in ad hoc networks and secure AODV,” *Ad Hoc Networks Journal (ADHOCNET)*, pp. 578–599, May 2007.

[57] A. Perrig, “Efficient collaborative key management protocols for secure autonomous group communication,” in *Proceedings of the Internationl Workshihp on Cryptographic Techniques and E-Commerce (CrypTEC’99)*, 1999, pp. 192–202. [Online]. Available: citeseer.ist.psu.edu/perrig99efficient.html

[58] S. Rafaeli and D. Hutchison, “A survey of key management for secure group communication,” *ACM Computing Surveys*, vol. 35, no. 3, pp. 309–329, Sept. 2003.

[59] S. Rafaeli, L. Mathy, and D. Hutchison, “EHBT: An efficient protocol for group key management,” *Lecture Notes in Computer Science*, vol. 2233, pp. 159+, 2001. [Online]. Available: citeseer.ist.psu.edu/rafaeli01ehbt.html

[60] A. Raniwala, K. Gopalan, and T. cker Chiueh, “Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 2, pp. 50–65, 2004.

[61] N. B. Salem and J.-P. Hubaux, “Securing wireless mesh networks,” *Special Issue of IEEE Wireless Communications Magazine on Wireless Mesh Networking, Theories, Protocols, and Systems*, vol. 13, no. 2, April 2006.

[62] S. Setia, S. Koussih, S. Jajodia, and E. Harder, “Kronos: A scalable group re-keying approach for secure multicast,” in *IEEE Symposium on Security and Privacy*, 2000, pp. 215–228. [Online]. Available: citeseer.ist.psu.edu/setia00krono.html

[63] N. Stakhanova, S. Basu, and J. Wong, “A taxonomy of intrusion response systems,” *International Journal of Information and Computer Security*, vol. 1, pp. 169–184, 2007.

[64] C. Strasburg, N. Stakhanova, S. Basu, and J. S. Wong, “Intrusion response cost assessment methodology,” in *ACM Symposium on InformAtion, Computer and Communications Security, ASIACCS 2009*, Sydney, Australia, March 2009.

[65] C. Strasburg, “A framework for cost-sensitive automated selection of intrusion response,” *Master Thesis*, 2009.

[66] Y. Sun, W. Trappe, and K. J. R. Liu, “A scalable multicast key management scheme for heterogeneous wireless networks,” *IEEE/ACM Trans. Netw.*, vol. 12, no. 4, pp. 653–666, 2004.

[67] P. Tague and R. Poovendran, "Modeling adaptive node capture attacks in multi-hop wireless networks," *ScienceDirect Ad Hoc Networks*, vol. 5, no. 6, pp. 801–814, August 2007.

[68] A. S. Tanenbaum, *Computer Networks*, 4th ed. Upper Saddle River, NJ: Prentice Hall PTR, 2003.

[69] G. Thamilarasu, S. Mishra, and R. Sridhar, "A cross-layer approach to detect jamming attacks in wireless ad hoc networks," in *Proceeding of Military Communications Conference MILCOM06*, Washington D.C, October 2006.

[70] G. Thamilarasu1, A. Balasubramanian, S. Mishra, and R. Sridhar, "A cross-layer based intrusion detection approach for wireless ad hoc networks," in *Proceedings of IEEE International Conference on Mobile Adhoc and Sensor Systems Conference (MASS)*, Washington, DC, November 2005.

[71] T. Toth and C. Kruegel, "Evaluating the impact of automated intrusion response mechanisms," in *the 18th Computer Security Applications Conference (ACSAC02)*, Las Vegas, NV, 2002, p. 301C310.

[72] C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt, "A specification-based intrusion detection system for aodv," in *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, 2003, pp. 125–134.

[73] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The versakey framework: Versatile group key management," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 9, pp. 1614–1631, 1999.

[74] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," *RFC 2627*, 1999.

[75] S.-H. Wang, C. H. Tseng, K. N. Levitt, and M. Bishop, "Cost-sensitive intrusion responses for mobile ad hoc networks," in *Recent Advances in Intrusion Detection (RAID) 2007*, 2007, pp. 127–145.

[76] X. Wang, J. Wong, and W. Zhang, "A heterogeneity-aware framework for group key management in wireless mesh networks," Istanbul, Turkey, September 2008.

[77] T. WiMAX Forum, *Available from http://www.wimaxforum.org/*.

[78] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[79] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *Networking, IEEE/ACM Transactions on*, vol. 8, pp. 16–30, Feburary 2000.

[80] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003, pp. 14–27.

[81] Y. Wu, B. Foo, Y. Mao, S. Bagchi, and E. Spafford, "Automated adaptive intrusion containment in systems of interacting services," *Computer Networks*, vol. 51, no. 5, pp. 1334–1360, 2007.

[82] Z. Yu, "Practical security scheme design for resource-constrained wireless networks," 2007.

[83] W. Zhang and G. Cao, "Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach," in *Proceeding of IEEE INFOCOM*, March 2005.

[84] Y. Zhang and Y. Fang, “ARSA: An attack-resilient security architecture for multihop wireless mesh networks,” *IEEE J. Select. Areas Communications*, vol. 24, no. 10, pp. 1916–1928, October 2006.

[85] Y. Zhang, W. Lee, and Y. Huang, “Intrusion detection technqiues for mobile wireless networks,” *ACM Wireless Networks*, vol. 9, no. 5, pp. 545–556, 2003.