THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

GROUP KEY GENERATION IN AD HOC WIRELESS NETWORKS USING A SUBGROUP METHOD

By

KRISTIN E. BURKE

A Thesis submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Spring Semester, 2003

The members of the Committee approve the thesis of Kristin E. Burke defended on April 25, 2003.

Alec Yasinsac
Professor Directing Thesis

Mike Burmester
Committee Member

Lois Hawkes
Committee Member

The Office of Graduate Studies has verified and approved the above named committee members.

ACKNOWLEDGEMENTS

I would like to thank Dr. Alec Yasinsac, my supervisor, for his many suggestions and constant support during this research. His patience was much appreciated and his persistence was inspirational. He deserves many thanks for always making himself available and for working so hard to motivate me when I needed it.

Also, I would like to thank Dr. Mike Burmester and Dr. Lois Hawkes for their time, energy and support for me with my thesis work. They were invaluable to me on my path to finding a thesis topic and developing as a researcher. I really appreciate all their hard work and consideration.

I am thankful, also, to the National Security Agency and the Department of Defense for the wonderful Information Assurance Scholarship, which allowed me to focus all of my free time on my thesis work. Thank you very much for choosing to support me and my efforts.

I am grateful to my mom for her patience and understanding, and to my dad for his mathematical brain. Many thanks to both of my parents who believed in me when I did not believe in myself.

Finally, I wish to thank the following: my sister Melissa (for her never-ending support and good ideas); Margaret, Melissa K. and Clare (for keeping me sane during some rough times); and The Security Group (for the great feedback and suggestions)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Most algorithms for Group Key distribution were not created with ad hoc networks in mind. In a wired network with a distinct infrastructure, generating a key distribution algorithm is based on a static environment and, therefore, more focus is placed on efficiency and security within those confines. With an ad hoc network, however, the focus must switch to the dynamic nature of the group and how to deal with the problems which it creates.

These problems, which include a lack of infrastructure and link failures between nodes, have been identified in a few protocols, including CLIQUES, the Burmester-Desmedt suite, and the YTCC protocols. These protocols, however, still need to rekey in a lengthy process when a link goes down within the group. Although AGKE has made this rekeying scalable, this process can still cause communication to be delayed for an extended period of time.

The solution to this problem proposed in this thesis is based on a subgroup method. This solution is based on a specific type of ad hoc network where the nodes are already organized into subgroups (such as a military operation, where nodes are broken into platoons). These subgroups choose a controller and are then organized as a tree based on the connections between them. Using this tree as a backbone, communication takes place between nodes in the group by using subgroup keys. This subgroup method also ensures that if a link is broken in the group, only that part of the tree needs to be re-keyed, providing a shorter down-time for the group.

CHAPTER 1

INTRODUCTION

Wireless networks have become an integral part of our society today. Whether they are instituted as the means for communication between cellular phones or as the avenue for contact between military agents on the battleground, wireless networks are an indispensable medium for correspondence.

Wireless networks are implemented in two flavors, ad hoc and structured. In a structured network, nodes can use invariable base stations to relay messages back and forth. This type of network is evinced in the cellular market, and provides security and reliability for that industry. Conversely, in an ad hoc network there is no set infrastructure, and nodes must communicate by routing each other's messages. Such networks are used for battlefield operations and rescue missions, where the nodes have little computational power and memory.

This non-structured characteristic of ad hoc networks makes them more vulnerable to attacks than structured networks, as the common method of communication, radio waves, are easily assailable through the use of the "right kind of radio"[1]. Thus, it is very important to establish keys for secure communication between the members of an ad hoc network to ensure eavesdroppers do not have access to sensitive material.

Group Key protocols provide a mechanism for providing this security in wireless networks. Instead of allowing messages to travel on these unsecured channels "free-and-clear", group key protocols provide a method for establishing a key amongst certain nodes who should receive the information, as opposed to those who just desire to receive it.

This paper discusses the security issues associated with ad hoc networks, the need for group keys, previous group key protocols and the problems with these protocols due to the nature of a specific group of ad hoc networks. This "specific group" refers to ad hoc networks that come complete with an underlying structure already in place, specifically a subgroup structure. An example of this type of network would be the military network that is already separated into platoons. This paper then proposes a new protocol, Group Key Generation Using Subgroups (GKGUS), for group key generation and distribution in these specific types of wireless ad hoc networks using a subgroup system based on the YTCC group key distribution protocol suite.

Does dividing these specific types of wireless ad hoc networks into subgroups improve the efficiency of group communication? The basis from which this question arose is offered in Chapter 2 and 3; Chapter 3 highlights other protocols in this area. The answer to this question is discovered in Chapters 4 through 6 and analyzed in Chapter 7. The final conclusions are discussed in Chapter 8.

CHAPTER 2

BACKGROUND

2.1 Ad Hoc Networks

Ad Hoc wireless networks are networks that have no fixed infrastructure (routers, bridges, gateways, etc). The nodes are mobile and therefore the links between them are dynamic. At any given time links may break and then reconnect in the same position, or in different parts of the network. Nodes of these networks function as both hosts and routers to relay messages back and forth between other nodes. Some examples of situations where ad hoc networks may arise include students desiring to interact in a lecture, a rescue operation after a natural disaster and military employments where operational information needs to be conveyed[2, 3]

2.1.1 Ad Hoc Network Characteristics

Ad hoc networks are most often characterized by their lack of structure. Unlike a wired network, or even a wireless network with an infrastructure (i.e. cellular networks), the nodes of an ad hoc network "dynamically establish routing among themselves to form their own network 'on the fly'"[4]. The networks themselves are usually temporary and the nodes must rely on each other to relay the messages to their correct destinations.

The nodes of the ad hoc networks are often bandwidth-constrained and energy-constrained as well[5]. This means that messages may be lost due to congestion and the actual throughput of the network is much lower than expected. Also, computations must be kept at a minimum and transmission range may be low due to lack of power.

Along with lack of structure, the nodes in an ad hoc network tend to move in and out of the range of other nodes quite frequently. This causes links to break and the network topology to change at any given moment. This is probably the most difficult characteristic to deal with when trying to produce routing and group key protocols.

2.1.1.1 Types of Ad Hoc Networks

There are different types of ad hoc networks that can evolve. One type of network is a random network, where nodes are not related to one another and have no semblance of

structure or order. There are other group key generation protocols for this type of ad hoc network, and this paper will not focus on that type of network.

Another type of ad hoc network is one where the nodes already have proximity partitioning and fall naturally into a hierarchical-tree type structure. This type of ad hoc network is one where the nodes are already split into small groups of nodes, or *subgroups*. Examples of this type of network include the military operation example, where nodes are broken up into platoons; and the rescue operation example, where nodes are broken up into rescue groups. The "structure" in this type of ad hoc network allows one to form protocols based on the existing organization of the nodes themselves.

2.1.2 Difficulties with Ad Hoc Networks

There are many difficulties associated with ad hoc networks. One of the main difficulties is link breakage, especially during route discovery and normal communication times for the nodes[6]. This lack of static topology also cause many problems in the group key area, which will be discussed in the next section.

Another problem in ad hoc networks is the lack of security for information being communicated within the network. Unlike a wired network, where it takes some work to be able to eavesdrop on communication, it is fairly easy to eavesdrop on wireless communications in a network with no set structure.

Lack of computational power is yet another ad hoc issue, and it has ramifications in both routing and group key protocols. In group key generation, lack of computational power yields to the importance of reducing exponentiations and other mathematically intensive operations.

2.2 Group Keys

Group Keys are the mechanism by which nodes in an ad hoc network communicate securely. If the information being passed in an ad hoc network may be received by anyone, then no group key is needed. The more likely case, however, is that group messages will be sensitive and, therefore, need a method for sending and receiving secured messages to other group members.

2.2.1 Group Key Generation

The challenge with developing group keys is that the nodes in the ad hoc network share no previous knowledge amongst them. Therefore, keys must be built upon virtually nothing; the protocols for group key generation must work from scratch to provide a secure key for the group.

There are several characteristics of group key generation that should be addressed. A group key may be either distributary or contributory, meaning that one group member creates the key or that all group members contribute to the key, respectively. While distributary key generation can make for a more efficient distribution and computation of the actual key, it has drawbacks. The first is that the node which is creating and computing the key may be generating weak keys that lessen the security of group communication. The second is that the computational power of that node must be great. Distributary key generation, however, allows all users to create a portion of the key, and therefore reduces the risk of malicious behavior.

Group key distribution itself can be centralized or decentralized, meaning that one node does the generation and distribution, or the work is spread out among a number of nodes. This reduces the chance of one node creating weak keys and lessens the computational weight on one node.

Group keys may also be based on authenticated members or non-authenticated members. Authenticated group keys use public keys or a comparable method to ensure that potential group members are in fact who they profess to be. This ensures that communication is even more secure because there are no members who are part of the group (and sharing the group key) who should not be group members.

CHAPTER 3

SIMILARITIES TO OTHER PROTOCOLS

There has been considerable development in the area of group key generation [7, 8, 9, 10, 11, 12, 13, 14], much of which focuses on ad hoc wireless networks (or are the basis for others which do). The proposed protocol has similarities to each of these, in one way or another.

3.1 Tree-based Protocols

There are several tree-based protocols that have already been developed for ad hoc networks. The Tree Based System of Burmester-Desmedt[8], Tree-Based Group Diffie-Hellman[12] and STR[14].

In the Burmester-Desmedt Tree Based System, nodes use blinded keys, a controller and a tree-format to generate a group key. Although robust and secure, a group based on this system must be re-keyed when a node leaves or enters the group. The TGHD (Tree-Based Group Diffie-Hellman) [12] approach is similar to the Burmester-Desmedt Tree Based System and attempts to distribute key generation and reduce the need for re-keying upon node enter or leave, however, there several instances have been noted where the entire group must be rekeyed upon node enter or leave. However, this is a logically-based tree, and therefore does not take advantage of the location of the nodes within the network. STR[15] is another protocol that attempted to make TGHD more simple, fault-tolerant and secure (with key independence), but remained inefficient. STR-improved[14] is an improvement over the original STR, providing a maximum of two rounds and two broadcasts per communication. It is noted, however, that even STR-improved, when implemented in a large group, can have extensive re-keying in some situations upon node leave.

3.2 Blinded-key Protocols

CLIQUES[10] is an example of a blinded-key protocol, where nodes use both their privately generated values (x_i) and the blinded counterparts (g^{x_i}) to generate group keys. Nodes are organized into a specific order and then each node passes its blinded key along with the portions of the other nodes' blinded keys raised to its secret random number. At the end of this process, the last node will broadcast portions of the actual key (generated

by raising a Diffie-Hellman generator to all private portions of all of the nodes one after the other) to every node so that all may compute the key. While secure by the Diffie-Hellman principle, this protocol requires extensive computations when rekeying is necessary and the nodes must be sequenced into a specific order.

Burmester-Desmedt[8] describes two blinded-key protocols, namely the "Star Based" and "Broadcast" systems. Both of these systems use blinded keys in three steps to generate a secure group key (by the Diffie-Hellman principle). The Star Based System has each node in the group send its blinded key to the group controller, who then computes the group key and communicates that key to the rest of the group securely. The Broadcast System is also a blinded-key system where all nodes send their blinded keys to each other and no group controller is needed to compute the key. These key generation systems provide security with authentication, but they are not scalable; as the group gets larger, the rekeying of the group when a node leaves becomes computationally intensive. Authenticated Group Key Exchange in Constant Rounds [13],is based on the Burmester-Desmedt Broadcast System protocol, but provides scalable authenticated group key distribution. Although rekeying is still required with every node enter/leave, this protocol has reduced the overhead of the overall keying operation.

3.3 The YTCC Protocol

GKGUS is most similar to the YTCC[9] protocol. The YTCC protocol is a highly efficient, robust and fully-distributed two-round message protocol that works in the following way:

1. One member announces formation of a group
2. The potential members (i= 1...n) select and publish a coordinator (member number 0) and the Diffie-Hellman base g and modulus p
3. Each i^{th} member (except the coordinator) chooses a secret random x_i and broadcasts its public g^{r_i}
4. The coordinator generates the random numbers z and x_0, g^{r_0}, $g^{r_i x_0}$ for each i, and encrypts e[z]$g^{r_i x_0}$ for each i. This coordinator then concatenates g^{r_0} with all the encrypted values and broadcasts.
5. Upon receipt of the broadcast, each member computes $g^{r_i x_0}$ using its private x_i, decrypts z, and computes a combining function $F = f(g^{r_1}, g^{r_2}, ..., g^{r_i})$ and the group key $K = g^{F \circ z}$, where F and $\circ$ are functions.

One similarity of GKGUS to YPCC is using the blinded keys of group members to ensure that the key is contributory as opposed to distributary. Another similarity is having a controller that generates the random part of the secret key and sends that part to other nodes. Also, the key generation is very similar in both protocols.

The proposed protocol here, however, optimizes the YTCC protocol, by reducing the need for an encryption mechanism to encrypt the random part of the secret key and the computational stress put on one controller node. Also, this protocol attempts to provide Key Independence and reduce the computational stress of re-keying on node entry and leave.

CHAPTER 4

GROUP KEY GENERATION USING SUBGROUPS

Although the definition of ad hoc specifies a property of having no structure, the author will argue that it is preferable to model the structure of the network after the underlying structure of the nodes for which it facilitates communication. Group Key Generation Using Subgroups (GKGUS) was developed specifically for ad hoc networks in which nodes are naturally divided into distinct groups, but where these groups must, at times, communicate with each other in the fashion of a larger, amalgamated group. Examples of this type of network include rescue operations, where searchers are broken up into groups to scour a particular area; military exercises, where soldiers are divided into platoons to cover an area; and disaster response, where groups may be partitioned by occupation (EMT, fire, police). These particular networks of nodes are already divided into subgroups simply by the natural order of the work that they perform. A network secured using GKGUS can be seen in Figure 4.1.

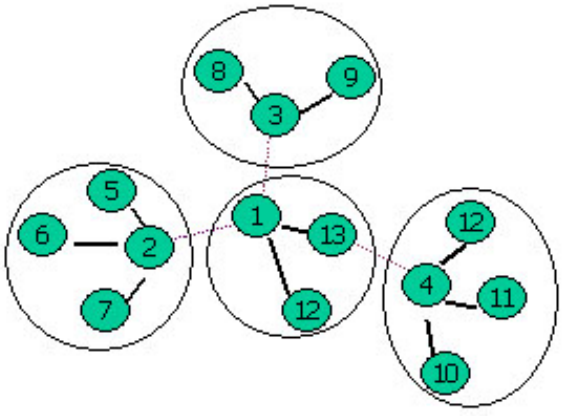

Figure 4.1. Cryptographically Secured Subgroups

4.1 Tree Generation

When a member signals that a group is to be created, the pre-existing subgroups must be organized into a logical tree structure based on their physical location. This tree-structure will be used to control the communication of the group.

The network itself must be organized into a tree so that nodes may be leaf nodes (with no children) or controller nodes (with children). There can be no cycles in this tree and every node must be included. This tree dictates how communication will be carried out within the network.

4.1.1 Subgroup Tree Generation Algorithm

Because of the specific type of ad hoc network being considered in this protocol, the nodes are already divided into subgroups physically. This makes generating the tree a rather trivial task, since the only links that actually need to be configured are those that connect the subgroups to one another.

To commence the Subgroup Tree Generation Algorithm (STGA), the subgroups must choose a controller amongst them. This controller must have have a connection to each of the other nodes in the subgroup, plus a connection to an outside node. If there is more than one node that satisfies this criteria, the controller is chosen randomly from these qualified nodes. If no node satisfies this criteria, a node is chosen with at least one link to another node in the subgroup, and a link to a node outside the subgroup. This process is shown in Figure 4.2(b). Controller nodes are marked with a *c*.

After this controller node is chosen, it serves as the parent node for all nodes within the subgroup with which it has a connection. If nodes are left over (the new controller did not have a connection to all nodes in the subgroup), the remaining nodes go through the same process as their own subgroup (choose a controller with the same properties, etc) until all nodes are included in the tree.

After the subgroups are organized, the nodes to which the subgroup controllers connect (the required "outside" nodes) then become parent nodes, and thus controllers as well. At this point, every node in all subgroups is connected into a larger group, as seen in Figure 4.2(c).

The root node is chosen as the controller node with the most links to other nodes in the group. If more than one controller node has the most links, the controller node closest to the middle of the entire group is chosen as the root. This may be accomplished through the use of GPS mapping or another location based service. If no such service is available, then the root node may be chosen at random from the nodes with the most links. The resulting tree is shown in Figure 4.2(d) with subgroups within the dashed circles.

4.2 Controllers

Any parent node in the tree structure is a controller. These controllers will generate the random values that serve to create the keys for the subgroups. This idea of splitting up the control from the primary controller to several controllers with an equal amount of authority is a vital part of subgroup functionality.

Each of these controllers serve as the link between the subgroup and the rest of the group. The controller serves to decrypt information from the subgroup and re-encrypt that information in order to send it on to its parent node. This facilitates the ease of member addition and deletion within the group, and allows each subgroup to have secret communication amongst its members.

Controllers may control more than one subgroup, if the number of nodes in one subgroup grows too large and must be partitioned, or if it is more efficient to start a new subgroup when a new member arrives in the group. This idea is discussed further in the Member Addition section.

4.3 Key Generation

The subgroup protocol is based on generating symmetric keys through a random number generated by a controller and the blinded keys of the members who share the key. The key that is used to encrypt is the same key that is used to decrypt information.

Let p be a large prime number and g a generator of $\mathcal{Z}_p$, universally known and used among the group members, and suitable for Diffie-Hellman calculations. Among the group members it is assumed that each member and has generated a secret random value $x_i \in \mathcal{Z}_{p-1}(i = 1...n)$ and its inverse ${x_i}^{-1} \mathrm{mod}(p-1)$(using the Euclidean algorithm or another suitable method). This g has been raised to x_i to produce a non-secret $g^{x_i} \mathrm{mod}\ p$ for each group member. These are referred to as *blinded keys*.

Each controller selects a random value $z_j \in \mathcal{Z}_{p-1}(j = 1..n_c)$, n_c being the number of controllers, and generates the subgroup key by calculating

$$h(f(g^{x_i}...g^{x_{n_s}}), g^{z_j}) \tag{4.1}$$

where $g^{xi}...g^{x_{n_{sg}}}$, n_{sg} being the number of nodes in the particular subgroup, are the blinded keys of all the nodes in the subgroup. This use of a function for combining of blinded keys succeeds in making the key contributory amongst the subgroup members. Both functions h and f are discussed further in the next section.

This controller then raises each of its children's blinded keys to its random z_j, generating $g^{x_i z_j}$ for each child. The controller then sends these values to its children in order for the children to calculate the subgroup key. This process is shown in Figure 4.3 in a reduced form of the original tree. In this figure, in order to conserve space, *1...*4 are all the blinded keys

of the nodes of those subgroups and $z_1 ... z_4$ are the random values created by each controller. This does not mean to imply that there is any sequencing in the random number generation between controlling nodes.

Upon receipt of this value, the child uses the inverse of its key, x_i^{-1} to derive z_j from the received value. After extracting this number, the child nodes can then compute the subgroup key using Equation (4.1).

The subgroups are cryptographically secured. Each subgroup has a controller and these controllers control their own subgroups comprised of their children and themselves. All controlling nodes, except for the primary root at the top of the tree, belong to two subgroups. This allows group communication to take place, which is discussed in the next section.

4.4 Key Generation Functions

Though the Key Generation Equation 4.1 is not very complex, the actual choice for h and f can make a notable difference in the security of the protocol itself. Encryption mechanisms that must be considered for the two functions include XOR, Multiplication, Exponentiation, DES, and hashing function.

4.4.1 Function *f*

There are several choices for the function f which will combine all of the blinded keys of the subgroup nodes in a specific subgroup to ensure that the key generated is contributory. There is no real emphasis on this function to increase the security of the information enclosed, because the information is itself public, however one may desire to have certain properties evolve from this function, i.e. randomness of output. Therefore, this function should be efficient and easy to compute, while still producing these properties. Choices for function f are highlighted in Table 4.1.

4.4.2 Function *h*

Function h is used to add the random portion to the key to make sure that it is secret and secure. This is the most important function of the key generation equation, because, unlike function *f*, the information given to the function is the secret random information and, thus, is needed to ensure the security of the key.

4.4.3 Two Functions vs. One

There is also the possibility of just using one function instead of h and *f*. If one function was used, all of the blinded keys of the subgroup nodes plus the secret group key g^{z_i} would

Function f	Function h	Advantages	Disadvantages
XOR	XOR	Easy to compute, efficient	Not secure enough for h
Multiplication	XOR	" "	Not secure enough for h
XOR	Exponentiation	Better security for h	Not as easy to compute
XOR	DES	f is easy to compute	Req. DES mechanism; block division
Multiplication	DES	f is easy to compute	" "
XOR	Hash	" "	Risk of collisions with hash

Table 4.1. Function Analysis

be combined into one DES, Hash, XOR, etc function. It can be argued that it provides no less security for the key, since the entire input for function f is publicly known. The function for f however may provide more randomness for input into h, but any node who knows the public keys and the function f could in fact generate this randomness and use it in h. An f function is best used if h is a hash function to ensure that the same order of blinded keys is entering the hash. Otherwise, one function or two makes no difference in the security of the key.

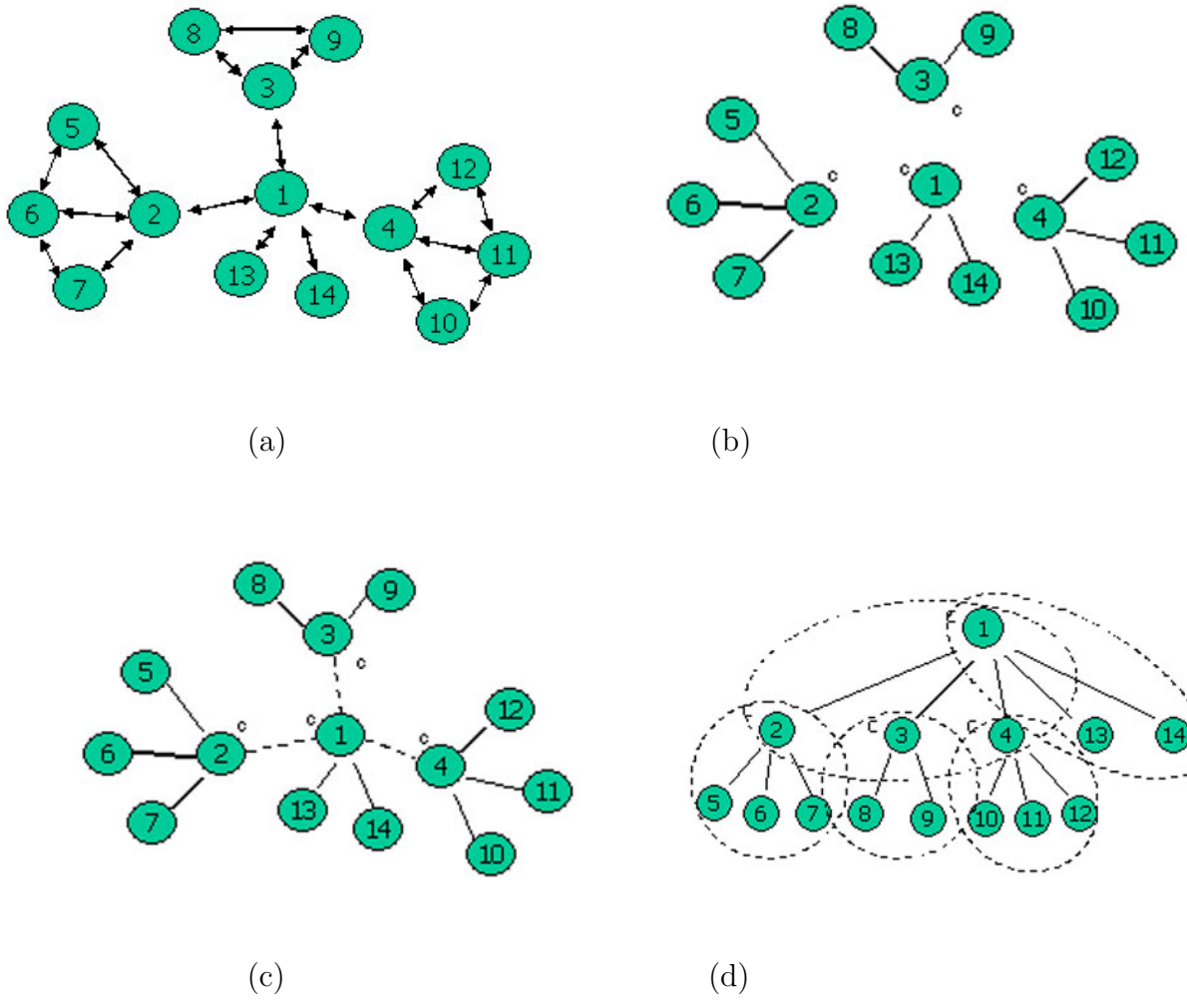

Figure 4.2. Subgroup Tree Generation: (a) original network, (b) subgroup connection , (c) full connection, and (d) resulting tree.

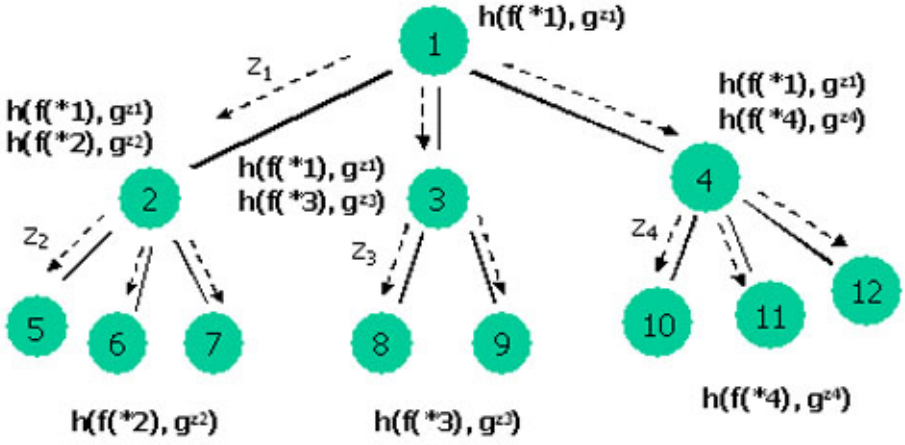

Figure 4.3. Key Generation Using Subgroups

CHAPTER 5

COMMUNICATION USING SUBGROUPS

There are several different ways that a node may wish to communication within the group, including Node to Node (Inside Subgroup), Node to Node (Outside Subgroup), Subgroup Communication and Group Communication. Based on the actual properties of the networks which are addressed in this paper, the assumption is that most communication will take place within the subgroup; however, all four modes of communication are explained in the following sections.

5.1 Node to Node Communication (Inside Subgroup)

When a node desires to communicate with another node within the same subgroup, it encrypts the message, addresses it for subgroup receipt and sends it to the controller. The controller receives the message and sends to the member of the subgroup to which it is addressed. The subgroup member then decrypts the message and receives the data securely. This process is shown in Figure 5.1.

5.2 Node to Node Communication (Outside Subgroup)

Each time a node wishes to communicate with another node outside its subgroup, it encrypts the message with its subgroup key, addresses it to the particular node and sends the encrypted message to its group controller. The subgroup controller then notices that the message is going outside of the subgroup and decrypts the message in order to re-encrypt it with its parent subgroup key. The subgroup controller then sends the message onto its parent subgroup controller node to be sent on to the destination node.

Upon receipt of this message, each controller node will decrypt and re-encrypt this message until the subgroup controller of the destination node is reached. When this occurs, that controller will simply re-encrypt with its subgroup key and forward the message to the destination node. This process is shown in Figure 5.2.

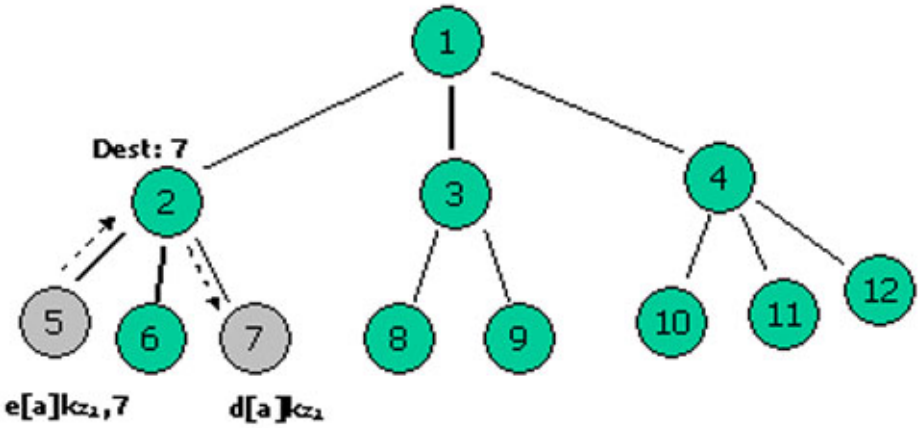

Figure 5.1. Node to Node Communication (inside subgroup)

5.3 Subgroup Communication

In order for the subgroup to communicate with one another, each node encrypts the message it wishes to send with the subgroup key, specifying no destination address, and multicasts the message to all other group members within its range. Each other subgroup member will multicast the message until all members have received the message. Upon receipt of the message, each node decrypts the encrypted message and receives the information.

5.4 Group Communication

Group communication takes place when a node sends a message to its subgroup controller addressed to the entire group. The subgroup controller then both multicasts the message to its subgroup and decrypts the message and re-encrypts with its parent subgroup key and sends the message to its parent node. The parent node (and all other controlling nodes) repeats this process until all nodes in the group have received the message.

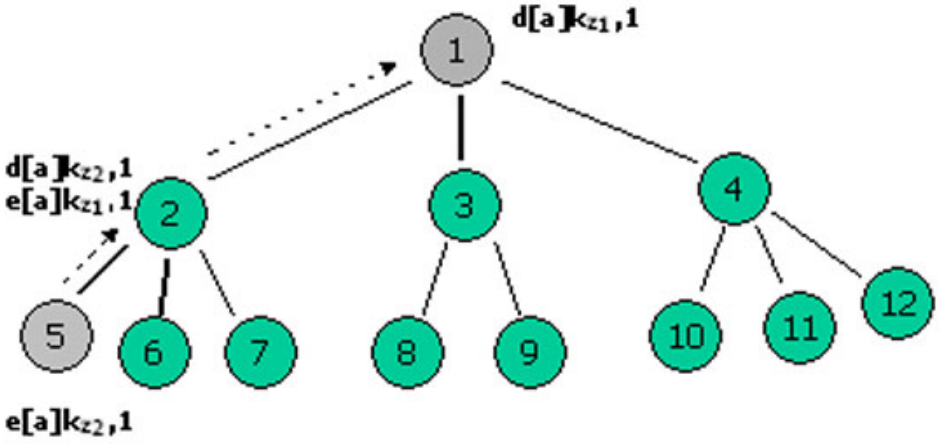

Figure 5.2. Node to Node Communication (outside subgroup)

CHAPTER 6

GROUP MEMBERSHIP

Group membership in an ad hoc network is a key component of a group key generation protocol. As members of the group leave and new members join, it is necessary to change the group key to make sure that the new members cannot have access to previously sent messages, and to ensure that old group members are not still involved in group communication.

Subgroups help with this matter by splitting the larger group into subgroups and therefore making the group membership more manageable and reducing (and sometimes eliminating) the lag time for re-keying. Due to the nature of the subgroups, when a member is added or deleted from one subgroup, all other subgroups may continue communicating with each other and the rest of the group.

6.1 Member Addition

A member joins the group by sending a join message, including the new member's blinded random $x_{new} \in \mathcal{Z}_{p-1}$ in the form of $g^{x_{new}} \text{mod} p$, to its nearest neighbor within the group. This join message will be forwarded to the nearest controller, who forces this neighbor node to become a controller and thus become the parent node to the new member. This is shown in Figure 6.1.

Node number 15 is attempting to join the group as a neighbor to node number 7. Node 7 is then forced to become a controller. This node then creates a new z_j, which is a random number to ensure key independence. After creating this random number, Node 7 generates the new subgroup key by using Equation (4.1) and sends the new portion of the key in the form of $g^{x_{15}z_j} \text{mod } p$ to the new member. This member then calculates the secret portion of they key, g^{z_j} and generates the subgroup key shared with Node 7 using Equation (4.1).

If a member joins at a node that is already a controller, that controller can rekey its entire subgroup to allow the node to join that subgroup. A threshold limit is set so that if too many members are joining at a certain node, that node may become the controller of more than one subgroup. Therefore, if that controller loses a child, it may simply rekey a portion of its children and reduce rekeying lag time again.

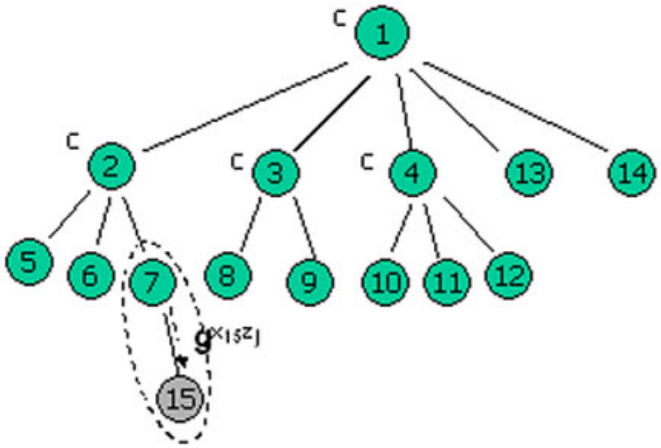

Figure 6.1. Member Addition

6.2 Member Leave/Partition

A member may leave the group by sending a leave message to its controller. If that controller has other children, as shown in Figure 6.2, it will create a new random value and re-key the subgroup using the method describe in section 6.1. Otherwise, the node will simply be decommissioned as a controller and no other action need be taken.

If the leaving node is a controller, then its children use their other current neighbors to search out another path to join the rest of the tree. If such a path is found, the children may be added to the tree by the member addition protocol. If no such path is found, the children are broken off from contact with the group and thus have formed a partition.

The partition may continue to function on its own using the current subgroup key, or may choose to create a new controller and rekey itself. Re-keying is a better option, considering the node that left the group still has the subgroup key, but if the controlling node simply failed, using the subgroup key would still be a viable option.

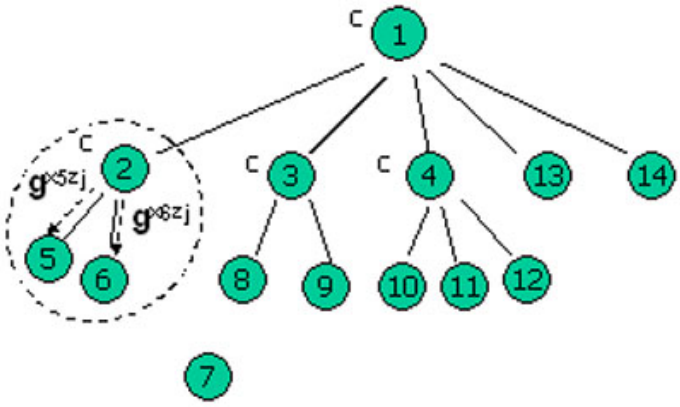

Figure 6.2. Member Leave

CHAPTER 7

ANALYSIS

There are many advantages and some disadvantages to using the subgroup method. Among the advantages are Key Independence, the efficiency of the protocol, and the dispersion of control. Disadvantages include the overhead of building the tree initially and the possibility of an insider attack by collusion.

7.1 Advantages

Advantages of the subgroup method stem from the tree-structure that organizes the subgroups. This tree-structure allows members to enter and leave efficiently, which is a very important property of this protocol. Also, organizing the nodes into subgroups follows the natural order of several types of ad hoc networks, which will be discussed in the next section. Providing keys for subgroup communication allows for secret communication in the large group as well as the smaller groups within.

7.1.1 Follows Natural Order of Ad Hoc Network

Although the definition of an ad hoc network specifies that they have no structure, one could argue that networks that are completely without structure are useless. In truth, the nodes within the network do have some semblance of order. For instance, take the oft-used military communication example. Although there is no infrastructure with which to communicate in the field and thus an ad hoc network is necessary for communication, one can argue that the division of military forces into platoons is structure in a ad hoc environment. These platoons are even examples of subgroups within an ad hoc network, thus evincing the fact that the subgroup method does follow the natural order of ad hoc networks.

Other examples of "structured" ad hoc networks include rescue missions, where rescuers are divided up into smaller rescue groups to facilitate coverage of large areas. The subgroup method allows these rescuers to communicate with each other and with the larger group based on the organization already in place. During a natural disaster or large-scale emergency, workers may be divided by their occupation, such as EMTs, fire department personnel and

police. These groups of people would be very likely to have private communication amongst themselves, but also need to be able to communicate with other groups of workers. Again, the subgroup structure is already underlying in the ad hoc network.

7.1.2 Secret Communication within Subgroup

There are times when it may be useful to have an avenue for secret communication within the subgroup itself. For instance, in the occupation subgroups described above, the police workers may want to discuss confidential subject matter amongst themselves, while still being able to communicate with other workers in the fire department, if necessary. The subgroup method facilitates this split in communication without the police workers needing to create another key for themselves outside of the protocol.

7.1.3 Efficiency Upon Node Enter

Upon entering the group, a node is attached to the tree where it is physically located. This aids in keeping the subgroups organized and keeping the nodes closest to the outskirts of the network to the outside of the tree. The new node comes into the range of a node already in the group and sends a join message (with its blinded random x^i) to that member. The member then becomes a controller node, and generates a random z^i to send to the new member. The new member can now generate the new subgroup key.

If the new node joins the tree at a node that is already a controller, that controller can choose at that point to start a new subgroup with just that node, or to rekey its entire subgroup including the new node. If the threshold of children is already met, however, the controller node must begin a new subgroup with the new node.

7.1.4 Efficiency Upon Node Leave

When a node leaves the group, the protocol is very efficient. As opposed to having one controlling node re-key the entire group, the subgroup controller must re-key and only if it has other children. If the controller has no other children in that subgroup, no action needs to be executed at all. Also, because of the threshold placed on how large subgroups can be, these re-keying calculations will reach a maximum and remain there.

If a controller leaves the group, the efficiency of the protocol wanes, but because the tree emulates the physical structure of the network, it is less likely that a controlling node will actually leave the group by moving out of the range of transmission from other nodes. Controlling nodes are the "inside" nodes of the group, and thus they are less likely to leave and cause a partition.

7.1.5 Physical Tree

Using a tree that represents the physical nature of the network, as opposed to a logical tree, has benefits as well. In representing the physical nature of the network, the nodes that are most likely to leave the network are at the outskirts of the network and thus become leaves in the tree. Therefore, when a node leaves the likelihood is that the node will be a leaf on the tree and will cost less than a node that a controller or higher on the tree itself.

7.1.6 Key Independence

Forward Secrecy, which is a fundamental security property in group key agreement, requires that a passive adversary (most likely a former group member) cannot discover new group keys based on the old keys. Backward Secrecy, also a fundamental security property, states the opposite; a passive adversary (most likely a new group member) cannot discover old group keys based on the new keys. Key Independence is the combination of Backward and Forward Secrecy.[16]

This protocol provides Key Independence for all keys generated with the Key Generation process described in section 4.4. Key Independence is reached because each z_i is random, and not based on any past z_i. Therefore, both forward secrecy and backward secrecy are possible because no adversary could generate any other knowledge of the key except for its current state.

This Key Independence is based on the assumption that the controllers will generate truly "random" z_i's as opposed to generating the same z_i's for different subgroups or basing future z_i's on previous ones. This assumption must be in place to provide both forward and backward secrecy, and thus Key Independence.

7.1.7 Dispersion of Control

The dispersion of control from one controller to several leads to the prevention of one node generating non-random or even the same random number over and over, thereby generating weak keys. As it is highly unlikely that all controlling nodes would be able to collude, and the fact that new nodes become controllers every time a node enters the group as a neighbor to them, this dispersion of control allows the group to communicate with little threat of an insider attack.

7.2 Disadvantages

While they are few, there are some disadvantages to using the subgroup method. The most offensive disadvantage lies in the original setup of the tree.

7.2.1 Building the Tree

The tree structure has many advantages, none of which is the original setup. It is difficult to make the tree because the nodes must know who their neighbors are and to some extent know who their neighbors communicate with. Although the subgroups are already formed naturally, producing the tree is a rather taxing process. While this can be noted as a disadvantage in a general purpose ad hoc network, this paper is discussing specific types of ad hoc networks where subgroups are already formed in the actual nodes of the networks, which would thus reduce the complexity.

Another benefit that clouds this disadvantage is the fact that once the tree is setup, it is basically self-maintaining. Unless there is some major division or partition in the group, the tree remains intact and nodes may add or remove themselves as they wish. Even when a controlling node leaves, its children can still maneuver back into the tree fairly easily or simply stay with a subgroup and never rejoin.

CHAPTER 8

CONCLUSIONS

The question proposed at the beginning of this paper speculated as to whether or not dividing specific types of ad hoc networks into subgroup improves the efficiency of group communication. There are many reasons why the answer to this question has been proven to be in the affirmative.

Firstly, the subgroup format, when used on specific types of ad hoc networks, provides a secure and efficient way for groups of nodes to communicate without having to interfere with the rest of the larger group's communication. The nodes encrypt with one key and send the messages to the controller, who then forwards it to the rest of the group. This is reliable and efficient.

Secondly, this format allows for efficient node enter and leave. When a node leaves from a certain subgroup, the rest of the nodes in the larger group need not be re-keyed. Only the subgroup needs to be rekeyed, and that is only if the controller has other children. If there were only two nodes in the subgroup, no action need be taken at all. As for a node entering the group, a random number needs to be generated for that node (or a subgroup according to threshold rules) and then keys will be regenerated. This is much more efficient than trying to rekey an entire group of greater than one hundred nodes.

Thirdly, the subgroup tree format allows the nodes that are furthest from the center of the network to be at the leaves of the tree. This means that the nodes that are most likely to leave the group create the fewest computations for the rest of the nodes upon departure. This keeps the controller nodes, the most important nodes, to the center and the most dynamic nodes to the leaves.

There are many other benefits of Group Key Generation Using Subgroups, as explained in Chapter 7, but the above benefits do, in fact, improve the efficiency of the communication of these specific types of ad hoc networks, and therefore gives a positive answer to the question proposed.

REFERENCES

[1] Stephen H. Wildstrom. Easy eavesdropping on wireless networks. *BusinessWeek Online*, February 2001.

[2] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *IEEE Personal Communications*, pages 46–55, Apr. 1999.

[3] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[4] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[5] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. Memo for the Internet Community, RFC2501, January 1999.

[6] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 66–75, 1998.

[7] Mike Burmester and Yvo Desmedt. Efficient and secure conference-key distribution. In M. Lomas, editor, *Security Protocols Workshop*, pages 119–129, Cambridge, UK, 1996. Lecture Notes in Computer Science 1189.

[8] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. De Santis, editor, *Advances in Cryptology - Eurocrypt '94*, Lecture Notes in Computer Science 950, pages 275–286, Springer, Berlin, 1995.

[9] Alec Yasinsac, Vik Thakur, Stephen Carter, and Ilkay Cubukccu. A family of protocols for group key generation in ad hoc networks. In *IASTED International Conference on Communications and Computer Networks*, 2002.

[10] Michael Steiner, Gene Tsudik, and Michael Waidner. CLIQUES: A new approach to group key agreement. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pages 380–387, Amsterdam, 1998. IEEE Computer Society Press.

[11] E.R. Anton and O.C. Duarte. Group key establishment in wireless ad hoc networks. In *E.R. Workshop en Qualidade de Serviço e Mobilidade*, 2002.

[12] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-based group key agreement. Technical Report 2002/009, 2002.

[13] Jonathan Katz and Moti Yung. Authenticated group key exchange in constant rounds. To appear, 2003.

[14] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Communication-efficient group key agreement. In *Proceedings of IFIP SEC 2001*, 2001.

[15] W. Diffie D. Steer, L. Strawczynski and M. Wiener. STR (a secure audio conference system). In *Crypto '88*, 1988.

[16] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *ACM Conference on Computer and Communications Security*, pages 235–244, 2000.

BIOGRAPHICAL SKETCH

Kristin E. Burke

Kristin Burke was born in Taunton, MA on February 17, 1978. She moved from Massachusetts at the age of 3 and grew up in Port St. Lucie, FL. In high school, she was involved in many extracurricular activities, including swimming, tennis, cheerleading and National Honor Society.

In the summer before her freshman year of college, Kristin carried the Olympic Torch for the Atlanta Games in 1996. She then began Florida State University in the Fall of 1996 as a Computer Science Major. After changing her major several times beginning in the Spring of 1997, Kristin finally returned to Computer Science in the Spring of 2000.

After graduating with a Bachelor's degree in Computer Science in December of 2001, Kristin then decided to get her Master's degree in Information Security at Florida State. She received the NSA/DoD Information Assurance Scholarship in Fall of 2002 and will be working for the Department of Defense upon graduation.