

Group Key Exchange Protocol Based on Diffie Hellman Technique in Ad hoc Network

Maryam Farajzadeh Zanjani

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2014
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Alexander Chefranov
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Alexander Chefranov _____
2. Assoc. Prof. Dr. Zeki Bayram _____
3. Asst. Prof. Dr. Gürcü Öz _____

ABSTRACT

During last decade, wireless ad hoc networks have been widely used for communication, transferring data or sharing some information for specific members. Nowadays security protocols play a fundamental role to provide a level of security for wireless local area networks (WLAN). Moreover, one of the most important issues to improve security by help of cryptography algorithms is generating a common key among participants to intercommunicate securely. The aim of thesis is creating a common secret key by means of Diffie Hellman (DH) technique, so the contributory group key exchange protocol is established in order to perform efficiently in context of ad hoc. To this aim, some analysis on Biswas's protocol (G. Biswas, IET Information Security, March 2008) and Tseng's protocol (Y.-M. TSENG and T.-Y. WU, INFORMATICA International Journal, April 2010) are done. Tseng's protocol fails to establish a common key in some situations, when the key generated by DH technique is not invertible. Thus, it is modified in order to fix the problem and achieve better performance in view of the computational cost for the proposed Tseng's modified protocol. Furthermore, theoretical analysis shows that computational cost in Tseng's modified protocol for each participant and the controller is decreased about 1.5 and 3 times in comparison with Tseng's protocol respectively. Tseng modified protocol is implemented and is tested for ad hoc WLAN with 3, 4, and 5 nodes.

Keywords: ad hoc, Wireless Local Area Network (WLAN), Network Security, Diffie Hellman Key Exchange (DH Key Exchange), Group Key Exchange (GKE), Tseng's protocol

ÖZ

Son yıllarda, kablosuz özel amaca yönelik ağlar iletişim, veri aktarımı veya bilgi paylaşımı için belirli kullanıcılar tarafından yaygın olarak kullanılmaktadır. Günümüzde güvenlik protokolleri, kablosuz yerel ağlarda (WLAN) güvenliği sağlamak için temel bir rol oynamaktadırlar. Ayrıca, şifreleme algoritmaları yardımıyla iletişimde güvenliğini artırmak için, kullanıcılar arasında ortak bir anahtar oluşturmak önemli konulardan biridir. Bu tezin amacı, Diffie Hellman(DH) tekniğini kullanarak gizli ortak bir anahtar yaratılmasıdır, böylece ad hoc ağlarda verimi artırmak için grup anahtar değiştirme protokolü oluşturulmuştur. Bu amaç için, Biswas protokolüne ve Tseng protokolüne bazı analizler yapılmıştır. DH tekniği ile üretilen anahtar tersi alınabilir olmadığı için, bazı durumlarda, Tseng protokolü ortak anahtar oluşturmada başarısız olur. Bu sorunu çözmek ve hesaplama maliyetini iyileştirmek için, Tseng protokolünün modifiyesi yapılmıştır. Ayrıca teorik analizler göstermiştir ki, modifiye edilen Tseng protokolü original Tseng protokolü ile karşılaştırıldığında, her bir katılımcı ve kontrolcü için, hesaplama maliyetinde 1.5 ve 3 kez azalma olduğu görülmüştür. Modifiye edilen Tseng protokolü 3, 4, ve 5 düğümden oluşan özel amaca yönelik kablosuz ağlarda test edilmiştir.

Anahtar Kelimeler: Özel amaca yönelik ağlar, Kablosuz Yerel Alan Ağı (WLAN), Ağ Güvenliği, Diffie Hellman Anahtar Değişimi (DH Key Exchange), Grup Anahtar Değişimi (GKE), Tseng Protokolü

Dedicated to my family

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my supervisor, dear Dr. Alexander Chefranov for the continuous support of my Master study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

Besides my supervisor, I would like to express my deepest gratitude and appreciation to Dr. Zeki Bayram and Dr. Gürcü Öz, for their encouragement, insightful comments, and remarks through the learning process of my Master degree.

Last, but not least, I am greatly indebted to my parents for supporting me spiritually throughout my life. I also would like to thank my dear friend Seyed Masoud Alavi Abhari for his constant encouragement and for hard working together before deadline.

TABLE OF CONTENTS

ABSTRACT	iii
ÖZ	iv
ACKNOWLEDGMENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xi
1 INTRODUCTION	1
2 DEFINITIONS AND RELATED WORKS	4
2.1 Definitions.....	5
2.1.1 Wireless Ad hoc Networks	5
2.1.2 Key Exchange Definitions.....	6
2.1.3 Diffie Hellman Key Exchange Protocol.....	7
2.1.4 Diffie Hellman Problems.....	8
2.1.5 Security Definitions.....	9
2.2 Related Work	12
2.2.1 Biswas's Group DH Protocol	12
2.2.2 Tseng's Group DH Protocol.....	14
2.3 Tseng's Protocol Analysis and Problem Definition.....	15
3 TSENG'S PROTOCOL MODIFICATION AND ITS SECURITY ANALYSIS.....	17
3.1 Tseng's Modified Protocol.....	18

3.2	Security Analysis	23
3.3	Performance Evaluation.....	28
4	IMPLEMENTATION AND EXPERIMENTAL RESULTS	35
4.1	Implementation	36
4.2	Experimental Results and Settings.....	39
5	CONCLUSION.....	46
	REFERENCES.....	47
	APPENDICES	51
	APPENDIX A: Programming Part	52
	A.1 Initialization.....	52
	A.2 Computation Process	55
	A.3 Printing the Results.....	58
	APPENDIX B: User Guide.....	59

LIST OF TABLES

Table 1: Comparison between Biswas Protocol, Tseng Protocol and Tseng's Modified Protocol	31
Table 2 : Initialization of Parameters	41
Table 3: TEXP (in millisecond) For Three Different Categories	42
Table 4: Average Messaging Time, Average Execution Time and Computational Cost (in millisecond) for 20 Runs of Tseng's Modified Protocol Based on Initialization Parameter in Table 2	44

LIST OF FIGURES

Figure 1: Diffie Hellman Key Exchange	8
Figure 2: Computational Cost of Controller for 25 Nodes	33
Figure 3: Modular Interface of Proposed Tseng Modified Protocol.....	36
Figure 4: Flowchart of Case 1	37
Figure 5: Flowchart of Case 2.....	38
Figure 6: Flowchart of Case 3.....	39
Figure 7 : Computational Cost Based On Experimental Result in Table 4 for Tseng's Modified Protocol	45

LIST OF ABBREVIATIONS

WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
GKE	Group Key Exchange
DH Protocol	Diffie Hellman Protocol

Chapter 1

INTRODUCTION

During last decades, communicating over an insecure public networks is widely discussed. The security and privacy of transmitted data without considering cryptography techniques in Wireless Local Area Network (WLAN) are compromised. It is clear that everyone can overhear in WLAN whether they are an adversary or not, thus it is necessary to consider some encryption algorithms to hide the plaintexts. In addition, one of the most important concepts in cryptographic algorithms is generating a shared key to communicate securely over a public channel.

Establishing a single group key for all members of a network can be a challenge from the point of view of ad hoc networks. While devices forming ad hoc networks are often mobile and low power participants also they often do not have much memory and computational power, the protocol should exchange the key as fast as possible. However, protocols that impose strong requirements for network topology are difficult to implement [1].

The specific and well-known method to generate the same secret key for both parties is Diffie Hellman (DH) key exchange [2]. It uses the exponential module, as the basis of its calculations also there is no need to transfer the shared key in communications, which is one of the most important features of DH key exchange. Moreover, there are so many protocols based on DH technique to create a common secret key among several parties [3]. It is obvious that the pieces of keys transferred

during the protocol should not reveal information that leads to the compromise of the group key.

The group key exchange protocols are divided into two categories; key agreement protocol, key distribution protocol. In this study, the emphasis is on key agreement or contributory, Diffie Hellman based protocols. In other words, all members of the ad hoc network should take equally part to establish a shared key.

There are two group key exchange protocols, which are discussed in this study; the protocol that is proposed by Biswas [4] and Tseng's group key exchange protocol [5]. The methodology of these protocols can be summarized in two steps. In first step, a DH key exchange is made between the controller node, which is a volunteer node, and other members. Then the controller uses the generated shared keys to establish a common secret key, also creates a message containing transformed shared keys, and broadcasts it in a network. Finally, members retrieve their own part from the given message to compute the common secret key.

While the protocols are often performed in context of ad hoc networks, the efficiency and flexibility of them should be considered. Tseng group key exchange protocol is not able to generate the key in some situations, thus a modification is needed on it. We propose Tseng's modified protocol to fix Tseng's protocol problem and also make it simpler and reduce the computational cost to achieve better performance. Moreover, security of the proposed Tseng's modified protocol is assessed.

In the end, not only the modified protocol compared with others protocol in terms of computational times and message sizes theoretically but also it is evaluated in

practice. Performance analysis and experimental results of Tseng's modified protocol are given to demonstrate that it is well suited for mobile devices with low computing capability in ad hoc or Wireless Sensor Networks. I will show that the modified protocol is a contributory group key exchange protocol and secure against the passive attack based on Diffie Hellman assumption [6].

This study is organized in four chapters. Chapter 1 is introduction. Chapter 2 is about definitions and some protocol related to group key exchange such as Tseng's protocol and Biswas's protocol. Chapter 3 provides Tseng's modified group DH protocol discussion and analyses the security of the protocol; it also compares the theoretical performance of Tseng's modified protocol and two mentioned group key exchange protocols, Biswas and Tseng's protocols. Moreover, Tseng's modified protocol is implemented and the modular interface of the protocol, implementation details and experimental results are demonstrated in Chapter 4. After finalizing this study in Conclusion, the whole view of developed codes, running procedure of the program, and the guidance to run the application are presented in Appendices.

Chapter 2

DEFINITIONS AND RELATED WORKS

In this chapter, ad hoc WLAN is discussed in Section 2.1.1 which is needed in order to prepare a context that all members of the network are capable of communicating with each other. Then in Section 2.1.2 some implications about key exchange definitions and different types of key exchange protocols in context of WLAN are explained. In addition, one of the most well-known key exchange protocol, Diffie Hellman Protocol, that is a basis for several group key exchange protocols is discussed in Section 2.1.3, also Section 2.1.4 is related to Diffie Hellman problems and assumptions. Moreover, Section 2.1.5 is about security definitions of the notion of a contributory key exchange protocol.

In Section 2.2 some Group DH key exchange protocols such as Hypercube and Octopus are introduced briefly and in Section 2.2.1 and Section 2.2.2, Biswas Group DH protocol and Tseng Group DH protocol are explained in details respectively.

Moreover, in Section 2.3, the needs for modification on Tseng Group DH protocol to perform as a contributory group key exchange and reduce the computational time are discussed and the problems are defined.

2.1 Definitions

It is clear that the security is needed for WLAN, which are more flexible and vulnerable than LAN. The initial security solution for wireless LAN relied on WEP (wired equivalent privacy) WEP used static keys in the encryption/decryption process to secure wireless communication. However, almost from the beginning, WEP was declared breakable and tools are readily available on the internet to break static keys [7]. Moreover, cryptography and encryption algorithms are used to protect network and data transition over WLAN or prevent possible threats. In the following section, a decentralized type of wireless network and its technical requirements are defined.

2.1.1 Wireless Ad hoc Networks

A wireless ad hoc network refers to any set of networks where all participants have equal status on a network and are free to communicate with any other ad hoc network members. The network is ad hoc because it does not have any pre-existing infrastructure. In other words, the connections are not through dedicated router. Instead, each node takes part in routing by sending data for others, so the decision to forwarding data from one point to another is made dynamically and definitely is due to network connectivity [8].

It is considered that the connections in ad hoc network are usually unreliable. They are often temporary networks and the participants can join or leave the network anytime. While the devices associated in ad hoc connections are often mobile, portable and do not have much computational power or memory, the contributing nodes may not be connected to the network for a long time. Thus, it is noticeable to consider these properties for establishing a shared key among all members of the network.

Nowadays there exist several algorithms and methods to provide security of WLAN, which depends on cryptographic methods. Definitely generating a proper key is one of the most important parts in encryption algorithms. However to apply any encryption algorithms it is needed that all nodes agreed on a shared key. In the following the key exchange definition and related issues will be discuss in details.

2.1.2 Key Exchange Definitions

Two-party Key Exchange protocol: The protocol is presented to the aim of establishing a session key between just two parties to encrypt/decrypt the transmitted data over an open and insecure network [9]. The best example for two-party key exchange is Diffie Hellman protocol, which uses two nodes to establish the secure shared key and is represented in Section 2.1.3.

Moreover, Group Key Exchange (GKE) protocol is designed to prepare a secure communication between a group (more than two) parties by establishing a secure shared key with the parties over an insecure channel [10].

Furthermore, role of the nodes that participate for producing the secure shared key should be considered. By raising the concept, protocols are classified in two categories.

The first category is key agreement Protocol also called Contributory Key agreement protocol [11] introduced to establish a secure shared key with participants when each of them equally plays their own specific role to produce the shared key. In other words, all participants in this way influence the outcome or they certify their part, which affect the shared key is up to date. In addition, no need of third party for producing the shared key is felt. Although in the Contributory key protocol nodes

have equal roles to generate the shared key, in Key distribution Protocol, the second category, just a party takes the duty for producing the secure shared key. Such that, the volunteer node autonomously, without taking the other parties into account generates the shared key, then distributes it to the other participants.

2.1.3 Diffie Hellman Key Exchange Protocol

Diffie Hellman key agreement is a specific method for exchanging keys. This method is one of the earliest and the most important foundations of implemented key exchange within cryptography field [12].

Diffie Hellman key exchange prepares a context for safe communication over an unsecure channel between two parties without having any prior knowledge from each other by sharing an agreed secret key. Moreover, the shared key will be used for symmetric encrypting the transmitted messages within the insecure channel.

This type of key agreement was first presented by Whitfield Diffie and Martin Hellman in 1976. In respect of Ralph Merkle's contribution to invention public-key cryptography, Martin Hellman named the algorithm Diffie–Hellman–Merkle key exchange.

Although Diffie–Hellman key agreement does not provide authentication in key exchange protocol, it prepares the foundation for many types of authenticated protocols.

The simplest and the original implementation of Diffie Hellman key exchange protocol is illustrated in the Figure 1. The protocol uses the multiplicative group of integers modulo P , where P is prime and α is a primitive root of P .

In the diagram Bob and Alice are going to communicate with each other, to prepare a secure shared key for communication in the insecure channel. Firstly, they should exchange the exponentiation with the base of an agreed value (α) with their own secret keys as exponents; all the operations are in modulo P . Secondly, Bob and Alice should calculate the received numbers to their own secret keys. Finally, the outcomes of the previous step should be mapped in modulo P . Although the base of the exponentiation in the first step is agreed on by Bob and Alice in advanced, but it may be public (even known to eavesdropper).

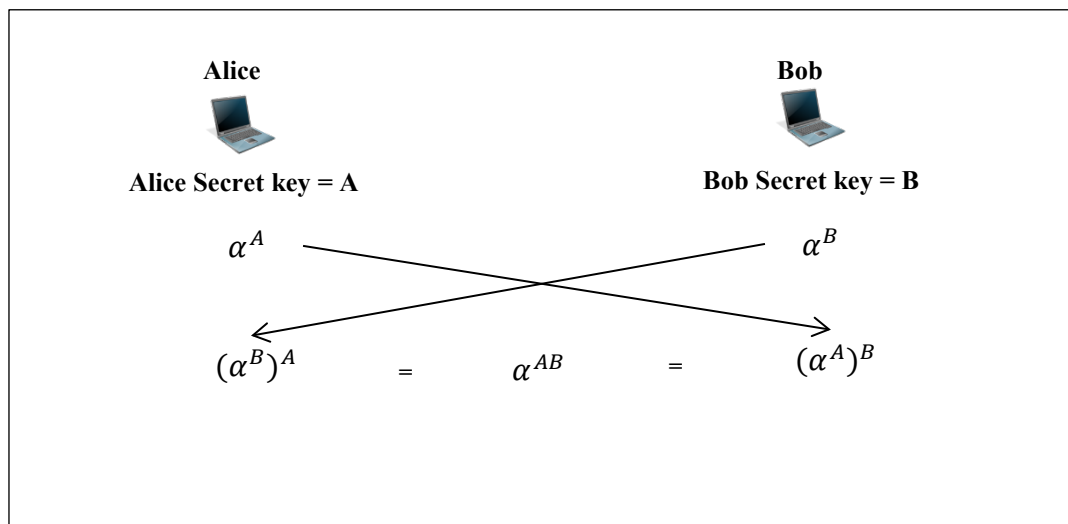


Figure 1: Diffie Hellman Key Exchange

2.1.4 Diffie Hellman Problems

One of the most important features of Diffie Hellman protocol is applying decryption without using the heavy computational reverse operation. Although, many mathematical operations of some security protocols work fast, the inverse operations such as decryption are hard to compute that is a motivation for the Diffie Hellman problem (DHP). DHP is a difficult mathematical problem. Moreover, if solving DHP were easy then an eavesdropper that observes α^A and α^B in Diffie Hellman key exchange, Figure 1, can compute α^{AB} easily and security is compromised.

Thus, in cryptography, Diffie Hellman problem is assumed hard for specific groups (where q is a prime number and g is a generator of the multiplicative group G of order q), also this assumption regularly named Diffie Hellman assumption. To the aim of difficulty of Diffie Hellman problem, three initial assumptions must be made. These three assumptions represented in follows:

a) Discrete Logarithm Assumption (DL):

The DL assumption is on how the eavesdropper can find x from given g^x when g is a member of group G while it is computationally difficult.

b) Computational Diffie Hellman Assumption (CDH):

The focus of the assumption to find g^{ab} from given g , g^a and g^b . In other words, the assumption states that by randomly chosen g , a and b from G for the tuple (g, g^a, g^b) , calculation of g^{ab} is computationally intractable [13].

c) Decisional Diffie–Hellman Assumption (DDH):

Decisional Diffie–Hellman Assumption is a foundation to prove security of many cryptographic algorithms. The aim of DDH Assumption is to state that given g , g^a , g^b and g^c , recognizing of two tuples such as (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) are computationally indistinguishable [14].

2.1.5 Security Definitions

This Section defines security properties that a protocol should consider. In a group key exchange system, which performs in context of ad hoc, participant nodes can communicate with each other by unicasting or broadcasting messages through networks. Obviously, a passive adversary or eavesdropper may gain the transmitted messages in public network and keep all previously communicated information.

However, the passive adversary cannot manipulate transmitted data or send modified messages to other participants. In the following, the security definitions for contributory group key exchange are introduced.

a) Group key exchange

Let GKE be a group key exchange protocol and assume that $G = \{G_1, G_2, \dots, G_n\}$ be group of volunteers wants to participate in the GKE protocol to generate a group shared key to communicate with each other.

b) Passive attack

In cryptosystem when a cryptanalyst could not interrelate with any other participant, he tries to influence and break the system by analyzing the observed transmitted data. This type of attack is called passive attack; moreover, it contains known plaintext attack while both plaintext and cipher text are exposed.

Passive attack is classified in two different types; the first type is Traffic Analysis, in that cryptanalyst foresees the treat of communication by detecting the frequency and length of transmitted message, finding out the position, analyzing the traffic and distinguish communicating hosts.

The second type is release of message contents. This type of attack monitors E-mail messages, conversation over telephone, chatting and transmitted files including personal and confidential data.

Actually, passive attack in group key exchange occurs when the cryptanalyst tries to detect the shared key by analyzing the some features (mentioned in two last

paragraphs) of transmitted messages or to discriminate against (distinguish between) the group key and a random bit string efficiently, over an open and insecure network.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. When the messages are exchanged neither the sender nor the receiver is aware that a third party has read the messages. This can be prevented by encryption of data [15].

c) Contributiveness

The third definition presents that participants cannot foresee the output of shared key on their own (individually). Thus, each party has a separate role for generating the group key. Moreover, each party can ensure the existence of its contribution to generate the common secret key.

d) Security in contributory GKE protocol

In this study, a contributory GKE protocol is secure when firstly contributiveness is provided for an existing group key exchange protocol such as GKE, secondly we can parry passive attacks of an assumed adversary A in contributory GKE protocol.

In this study, it is assumed that the group key exchange protocols are of non-authenticated type. However, the authentication can be achieved by considering some signature techniques. Obviously, in authenticated Group key exchange the active adversaries and the threats should be involved in security model when modification of a transmitted data compromise communications.

2.2 Related Work

There are several solutions for extending the Diffie Hellman key exchange to a group key agreement. Actually so many works have been proposed and the earliest (1982) one is by Ingermarson et al. [16]. The protocol assumes that it is allowed for the participants to form a ring due to the network topology. Another protocol was proposed by Steiner et al [17], it has some security risks. Furthermore, the Hypercube protocol [3] that is based on DH key exchange is vulnerable to node failure due to the strict requirements on network topology; also, the Octopus protocol [3] uses a hypercube in its center, defiantly inherits the vulnerabilities and the threats of the hypercube.

However, none of the protocols achieves the optimum efficiency values and they are not well suited for a changing network. In the following, two protocols that try to establish a session key dynamically for secured communication are discussed.

2.2.1 Biswas's Group DH Protocol

Biswas [4] proposed an efficient contributory multi-party key-exchanging technique for a large static group. In this protocol, which is based on Diffie Hellman technique, a member who acts as a group controller configures two-party groups with other participants and creates a DH-style shared key for each group; then combines these generated shared keys into a single multi-party key and behaves as a normal group member. It is assumed that two parties are agreed about two large positive integers; q and α . Considering, q is a prime number and α is a generator of a finite cyclic group G of order q . The protocol can be summarized in two steps.

Step 1: An arbitrary member acts as a group controller, for example P_c , and exchanges public keys with other members. Each group individually generates a DH-

style key using DH technique. Obviously, the public Key X_c for group controller P_c is generated using the DH formula as below.

$$X_c = \alpha^{e_c} \text{ mod } q \text{ (} e_c \text{ is a private key of controller)}$$

The public Key X_i for node P_i is generated using the formula:

$$X_i = \alpha^{e_i} \text{ mod } q \text{ (} e_i \text{ is a private key of the node)}$$

Each member similar to the basic DH generates a unique shared key, K_i with group controller as

$$K_i = \alpha^{e_i e_c} \text{ mod } q$$

Step 2: a group controller actually calculate $n - 1$ shared keys for $n - 1$ groups.

Then, it combines these generated keys to make a single Group key Y_i to send it to the node P_i

$$Y_i = \alpha^{\prod_{j=1}^{j=n} K_{j \neq i}} \text{ mod } q$$

On receiving, each node P_i produces the group key K as follows:

$$P_1 \text{ generates } K = (Y_1)^{K_1} \text{ mod } q = \alpha^{K_1 K_2 K_3, \dots, K_n} \text{ mod } q$$

$$P_2 \text{ generates } K = (Y_2)^{K_2} \text{ mod } q = \alpha^{K_1 K_2 K_3, \dots, K_n} \text{ mod } q$$

$$P_3 \text{ generates } K = (Y_3)^{K_3} \text{ mod } q = \alpha^{K_1 K_2 K_3, \dots, K_n} \text{ mod } q$$

...

$$P_n \text{ generates } K = (Y_n)^{K_n} \text{ mod } q = \alpha^{K_1 K_2 K_3, \dots, K_n} \text{ mod } q$$

While the group controller knows all shared keys so it generates the group key and becomes a usual member of a group:

$$K = \alpha^{K_1 K_2 K_3 \dots K_n} \text{ mod } q$$

It is noticeable that Biswas protocol has been compared with other multi-party key [18] [19] generating techniques, and the results obtained were better than previous mentioned protocols. Moreover, he claims that the contributiveness is present in his technique.

2.2.2 Tseng's Group DH Protocol

Tseng [5] expresses security weakness of Biswas's Group-DH protocol, also demonstrates that Biswas's protocol is not a contributory protocol because the controller node is able to predetermine group secret key by him/her. Therefore, he designed a group key exchange protocol. Indeed, Tseng's protocol is a development on Biswas's protocol and clearly based on the same Diffie–Hellman technique. By Tseng's protocol improvement the contributiveness of all members are verifiable. In other words, all participants can confirm their role for constructing a group secret key, by restoring their own part in order to generate the common group key. Moreover, in the view of passive attacks Tseng demonstrated that his protocol is secure. In the following Tseng's Group DH protocol is explained in details and summarized in two steps.

Step 1: The first step is similar to the Biswas protocol in Section 2.2.1. However, Tseng considers a pure prime number to achieve more security in definition of Diffie Hellman parameters [5]. It means that, he considers a large prime number as p and another large prime number q where $q=2p+1$ also the group G_p is a subgroup of

quadratic residues in Z_q^* that is $G_p = \{i^2 \mid i \in Z_q^*\}$. In addition, α is a generator for the subgroup G_p .

Step 2: The controller node P_c chooses a value randomly as x then it tries to compute the following.

$$Y = \alpha^x \bmod q \quad , \quad Y_i = Y^{K_i^{-1}} \bmod q \quad (1 \leq i \leq n-1)$$

Then, P_c broadcasts $(Y_1, Y_2, Y_3 \dots, Y_{n-1})$ to each participant node. Finally, each participant P_i can compute the group key:

$$K = H(Y_i^{K_i}, Y_1, Y_2, Y_3 \dots, Y_{n-1})$$

Actually, it is remarkable that each participant node should retrieve the amount of Y from the broadcasting message to calculate the common group key and be able to communicate with other nodes securely.

2.3 Tseng's Protocol Analysis and Problem Definition

The network topology of ad hoc can be changed rapidly, because each member node may decide to leave or join the network. While the ad hoc network's properties are determined in Section 2.1.1, it is extremely noticeable that portable devices taking part in these types of networks to communicate securely should establish a group key as fast as possible. Thus, the parameters such as computational cost and message sizes together with contributiveness and security are considerable to achieve the key exchange time as short as possible. Although, Tseng demonstrates that his group DH protocol is secure against passive attack, there exist some situations within establishing a shared key is not possible at all; when the multiplicative inverse of the

key does not exist and the protocol does not support these situations explicitly. Thus, a modification is considered as the proposed Tseng's modified protocol.

Moreover, in the view of performance, there is an attempt to reduce the computational cost of Group key exchange to be well suited for devices with low computational power and not large memory. While the battery consumption for proposed protocol is decreased due to less computational complexity, it can be suitable also for wireless sensor networks. The Tseng's modified protocol is proposed and explained in detail in next chapter; also, implementation and experimental results are represented in chapter 4 showing satisfactory results.

Chapter 3

TSENG'S PROTOCOL MODIFICATION AND ITS SECURITY ANALYSIS

The purpose of this chapter is explanation of the Tseng's modified protocol. The protocol is performed in context of ad hoc WLAN. Each participant is able to be a controller to establish a common secret key; also, each node can request the common group key by broadcasting its own public key. All nodes, same as in Tseng's protocol, which receive the request, send their public keys back to the requester (controller) node and a two party Diffie Hellman key exchange is performed to generate shared keys for each group. After computing the corresponding amount for each group, the controller broadcasts a message including the transformed keys for each related node. In the end, nodes that are received the message, retrieve their own part and use it to found the common group key.

Subsequently the details of the Tseng's modified protocol are discussed in three Sections. In Section 3.1, the methodology and the structure of modified protocol are addressed. Section 3.2 is about security analysis of the proposed protocol in the view of passive attacks and contributiveness. The last Section that is related performance evaluation aims to compare the computational cost and the message size of the Tseng modified protocol with Biswas and Tseng's protocols theoretically. In this study, Visual C#.Net has been applied as the programming language and the implementation and experimental results are demonstrated in Chapter 4.

3.1 Tseng's Modified Protocol

In this Section, a modification is proposed on Tseng's protocol [5] and the modified Tseng's protocol is described in details. It is clear that the system parameters that are used in modified protocol are similar to Tseng's one. It is also assumed that the neighboring nodes have already authenticated each other.

Considering the second step of the Tseng's Protocol as it is mentioned in Section 2.3, the group controller should compute the multiplicative inverse of the DH shared key for all participants to find the amount of $Y_i = Y^{K_i^{-1}} \bmod q$. While the idea is that each participants should be able to restore amount of Y based on Tseng's protocol that is

$$Y = (Y^{K^{-1}})^K \bmod q \quad (1)$$

The point is that, in this term modular multiplicative inverse of K that is K^{-1} does not always exist. Based on Euler Theorem that express for any integer α and prime number q , $\alpha^{\varphi(q)} \equiv 1 \pmod{q}$, $q \nmid \alpha$ Then, $\varphi(q) = q - 1$, so $\alpha^{q-1} \equiv 1 \pmod{q}$; Thus, to retrieve amount of Y from the message the following formula should be considered, when $Y^{A(q-1)} \bmod q$ is equal to 1 due to Euler Theorem.

$$Y = Y^{A(q-1)+1} \bmod q = (Y^{A(q-1)} \cdot Y^1) \bmod q = Y^1 \bmod q \quad (2)$$

Therefore, regarding to (1), (2) and Fermat's Little Theorem proved by Euler's Theorem [20], K^{-1} should be computed just same as the following.

$$K \cdot K^{-1} \bmod (q - 1) \equiv 1, \text{ where } q \text{ is a large prime number} \quad (3)$$

Moreover, in (3) the multiplicative invers of K exists if and only if Greatest Common Divisor of K and $q - 1$ is equal to one. In other words, $GCD(K, q - 1) = 1$. (4)

However, in Tseng's protocol in order to find the multiplicative inverse of K for each node, it is not explicitly denoted that modulo $(q - 1)$ should be considered. In addition, the amount of $K = a^x \bmod q$ is dependent on the amount of a random exponent that makes it hard to guess whether K is invertible or not.

Considering (3) and (4), while $q - 1$ is an even number, finding multiplicative inverse of K can be so challenging in Tseng's protocol. In other words, K has to be a co-prime with an even number such as $q - 1$ and defiantly less than q while $q = 2p + 1$. In this case selecting a proper K from the multiplicative group G is a difficult problem while, there may be existing some numbers such as $2, p$ or $2p$ that $q - 1$ can be divided by them. The possibility of choosing each number, as K from group G is as equal as others, whether the number is odd or even.

Thus, there is not any guarantee to compute a proper K . Moreover, if an appropriate K is not selected then the multiplicative inverse does not exist and leads to failure. Although, in the view of cryptography the prime number p is considered a large number that provides more opportunities to supply an appropriate values for invers modules; significantly the possibilities of failure is not low.

Here, there is an example that causes to fail. The system parameters are based on Tseng's protocol introduced in Section 2.2.2.

$$q = 23, p = 11, G_q = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\},$$

For the mentioned group α can be 2, 6 or 8. Here $\alpha = 6$.

Step One	
Controller C	Participant A
$e_c = 2$	$e_A = 3$
$X_C = \alpha^{e_c} \bmod q = 6^2 \bmod 23 = 13$	$X_A = \alpha^{e_A} \bmod q = 6^3 \bmod 23 = 9$
$K = X_A^{e_c} \bmod q = 9^2 \bmod 23 = 12$	$K = X_C^{e_A} \bmod q = 13^3 \bmod 23 = 12$
It is successfully done: $12 = 12$	
Step Two	
Controller C	Participant A
$Y = \alpha^x \bmod q = 6^4 \bmod 23 = 8$	
$K^{-1} \cdot K \bmod (q - 1) = 1$ $\rightarrow K^{-1} \cdot 12 \bmod 22 = 1$ $\rightarrow GCD(12, 22) = 2 \neq 1$ \rightarrow <i>it is not invertible</i>	
There is a failure: the key is not invertible.	

Considering group G_q and $(q - 1) = 22$, all even numbers are not invertible. In other words, for most of the G_q members it is not possible to compute K^{-1} modulo $(q - 1)$. The only proper choices are 3, 9 or 13; that is just three values from eleven possible values in G_q . Thus, the probability of failure for n participants is $1 - \left(1 - \frac{8}{11}\right)^{n-1} = 1 - (0.27)^{n-1}$. Obviously, as the numbers of participants are increasing, the probability of failure grows significantly.

Moreover, regarding to Tseng's protocol parameter discussed in Section 4.2.2, probability of failure for n nodes is defiantly more than the following:

$$P_{Failure} = 1 - \left(1 - \frac{\sqrt{q}/2}{p}\right)^{n-1} \quad (5)$$

Due to this formula (5), probability of failure in mentioned example should be greater than $1 - \left(1 - \frac{\sqrt{23}/2}{11}\right)^{n-1} = 1 - (0.78)^{n-1}$.

Hence, Tseng's protocol is non-deterministic algorithm that sometimes cannot establish a group key. It is remarkable that repeating the Tseng's protocol when a failure occurred may be a solution but absolutely requires more computation and it is time consuming especially for large number of participants. In addition, repeating Tseng's protocol may still cause to fail.

On the other hand, among the basic arithmetic operations, the computation of a multiplicative inverse is the most time consuming operation, Tseng considers modular inverse together with modular exponentiation to prepare the key message that takes so much time. In other words, both modular operations should be performed N times, where N is number of participant nodes; also in order to provide security, large numbers should be used as an input of these algorithms. While the running time of modular exponentiation is $O(\log(\text{exponent}))$ [21], for XOR operation it is $O(1)$ as it is just one operation that is performed for specific amount of data.

Thus, a modification is done in step 2 of the protocol to deal with the mentioned problem and decrease the time of running procedure to be well suited for low power devices in Wireless Sensor Networks. In Tseng's modified protocol, instead of computing and restoring Y due to modular exponentials and modular inverse, an

Exclusive Or (XOR) operation that is faster with less computational complexity is considered. Obviously, XOR is often used as a simple mixing function in cryptography. Considering XOR as a function, while both K and Y are as inputs of this function, also are not clear for others (an eavesdropper), it will be difficult to understand the result of the XOR function.

Furthermore, the proposed protocol is implemented in context of ad hoc network, the details of establishing a common secret key theoretically is presented in two steps as follows.

Step 1: This step is same as step 1 in Tseng's protocol.

Step 2: In the second step while the controller knows all DH shared keys, it selects a random value such as x and computes the amount of Y based on the following formula.

$$Y = \alpha^x \text{ mod } q$$

Then, the controller uses Y and K_i to calculate Y_i as below:

$$Y_i = Y \oplus K_i \quad (\text{Where } 1 \leq i \leq n-1) \quad (6)$$

In the end, the controller broadcasts $(Y_1, Y_2, Y_3, \dots, Y_{n-1})$ to each participant. Actually, each participant P_i and controller can find out the common secret key based on the following formula:

$$K = H(Y_i \oplus K_i, Y_1, Y_2, Y_3 \dots , Y_{n-1})$$

Thus, by replacing XOR operation with modular exponentiations, not only it is possible to establish a group key for all participants, but also the computational cost is reduced to achieve the short time as possible to generate the key. Moreover, the security analysis in next Section is provided to demonstrate that Tseng's modified protocol can be taken into account as a group key exchange protocol, which provides security and contributory.

3.2 Security Analysis

This Section involves some assumption to show that the Tseng's modified protocol is a secure group key agreement protocol. An obvious requirement for such a group is that it must provide safety against passive attacks; also, the participants should be convinced about their contribution in generating the key.

First of all the contributiveness of the modified protocol is proved under the one way hash function assumptions [22, 23]. These assumptions that are explained below, show that for a secure one-way hash function such as H,

$$H: S = \{0,1\}^* \rightarrow L = \{0,1\}^l$$

Considering l as a fixed length, the requirements mentioned below are satisfactory.

- a. For any $y \in L$ it should be difficult to detect any message as m while, $y = \text{hash}(m)$, considering $m \in S$. Actually, it is infeasible to generate a message that has a given hash.

- b. Given any message $m_1 \in S$ it should be hard to find another input $m_2 \in S$ such that $m_1 \neq m_2$ and, $\text{hash}(m_1) = \text{hash}(m_2)$. In other words, a modification on a message without changing the hash is infeasible.
- c. It should be difficult to find two different messages m_1 and m_2 such that they have the same hash; $\text{hash}(m_1) = \text{hash}(m_2)$.

Thus, under the mentioned requirements it can be concluded that if all participants can establish the common secret key then each of them can be sure that their part is included in generated group key. In addition, when the group controller broadcasts the final message, each member P_i has to find his/her part and use his/her shared key K_i , to restore the controller part Y from the message in order to compute the common secret key K . The following equations hold while the group key is established among participants for secure communication.

$$\begin{aligned}
 K &= H(Y_1 \oplus K_1, Y_1, Y_2, Y_3 \dots, Y_{n-1}) = H(Y_2 \oplus K_2, Y_1, Y_2, Y_3 \dots, Y_{n-1}) = \dots \\
 &= H(Y_{n-1} \oplus K_{n-1}, Y_1, Y_2, Y_3 \dots, Y_{n-1})
 \end{aligned}$$

Moreover, due to (6) in Section 3.1, there exists a value Y such that,

$$Y = Y_1 \oplus K_1 = Y_2 \oplus K_2 = \dots = Y_{n-1} \oplus K_{n-1}$$

Also each participant computes his own part,

$$\begin{aligned}
 Y_1 &= Y \oplus K_1 \quad , \\
 Y_2 &= Y \oplus K_2 \quad , \dots, \\
 Y_{n-1} &= Y \oplus K_{n-1}
 \end{aligned}$$

Thus, by replacing the mentioned computed amount of Y_i , in generated common group key that is $K = H(Y_i \oplus K_i, Y_1, Y_2, Y_3 \dots, Y_{n-1})$, the key can be computed as $K = H(Y, Y \oplus K_1, Y \oplus K_2, \dots, Y \oplus K_{n-1})$. It means that the key is produced based on the controller part Y and the participant contributiveness K_i . Therefore, K_i is needed to generate the key and since $K_i = \alpha^{e_i c_i} \text{ mod } q$ is made due to private key of the participants; it can be deduced that all nodes are assured to agree on the common group key.

On the other hand, Tseng's modified protocol is secure in the view of passive attacks by using XOR operations and based on Diffie Hellman assumptions. Considering Exclusive OR operation as an important basis of cryptography it has been commonly used in many complex cryptographic algorithms such as DES, AES and MD5 [24]. Therefore, simple XOR cipher can be used as a kind of additive cipher to encrypt and decrypt a plaintext symmetrically. The principles denote the XOR cipher:

$$\text{plaintext} \oplus \text{Key} = \text{ciphertext},$$

$$\text{ciphertext} \oplus \text{Key} = \text{plaintext}$$

Consider that \oplus points out the (XOR) operation. There is an example that the word "Wiki" (by using ASCII code of each word) is encrypted and then decrypted by the use of the key (11110011 11110011 11110011 11110011).

Encryption:

$$\begin{array}{cccccccc} 01010111 & 01101001 & 01101011 & 01101001 & \text{(Wiki)} & & & \\ \oplus & & & & & & & \\ 11110011 & 11110011 & 11110011 & 11110011 & \text{(Key)} & & & \\ \hline 10100100 & 10011010 & 10011000 & 10011010 & \text{(The cipher text)} & & & \end{array}$$

Decryption:

$$\begin{array}{cccccccc} 10100100 & 10011010 & 10011000 & 10011010 & \text{(The cipher text)} & & & \\ \oplus & & & & & & & \\ 11110011 & 11110011 & 11110011 & 11110011 & \text{(Key)} & & & \\ \hline 01010111 & 01101001 & 01101011 & 01101001 & \text{(Wiki)} & & & \end{array}$$

The XOR cipher will be much more secure than key repetition within a message using in some cryptographic algorithms, where the random key is greater or as long as the message [25]. Thus, the notion and the less computational time complexity of XOR inspire me to use Exclusive OR instead of exponentiation. Moreover, the probability of compromising the security is investigated as below. In the worst case, assume that eavesdropper has grabbed the cipher text and he found that the cipher came out from an operation (Exclusive OR) of two operands, therefore for each bits of the operands he would encounter two cases mentioned bellow.

Known for eavesdropper	Plaintext	Key	Case
0	0	0	1 st
	1	1	2 nd
1	0	1	1 st
	1	0	2 nd

Therefore, the probability that eavesdropper by knowing the bit from cipher text truly can guess the related bits of the plain text and the Key is equal $\frac{1}{2}$.

Moreover, where the cipher text contains n bits then the probability that the eavesdropper truly can guess the related bits of the plain text and the Key will be calculated as follows:

Cipher text	1	1	0	...	1
Plaintext	?	?	?	...	?
Key	?	?	?	...	?
Probability	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$...	$\frac{1}{2}$

Thus the complexity of XOR operations for discovering the key and the Plaintext is:

$$\underbrace{\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \dots \times \frac{1}{2}}_{n \text{ times}} = \frac{1}{2}^n = \frac{1}{2^n}$$

Hence, the complexity of XOR operation in order to discover the key, in modulo p that is a very large prime number will be $1 - (1/2^p)$ that is not a small value.

Furthermore, the security can be threatened if one of the inputs (key or plaintext) is known. However, in case of Tseng's modified protocol, the inputs are the shared key K_i and the amount of Y that both values are difficult to guess. In other words, for the discrete logarithm equation $Y = \alpha^x \bmod q$, which is generated and known just by controller, also it depends on the amount of random value x that is chosen by controller again, thus to find out Y is needed to know amount of x that is difficult.

Moreover, under the Decisional Diffie Hellman Assumption (DDH) discussed in Section 2.1.4 it is difficult to recognize K . It means that even if an eavesdropper gets $\alpha^C \bmod q$ that controller broadcasts to network and $\alpha^A \bmod q$ that calculated by participant A, also can guess a random value such as R in group G as $R = \alpha^z \bmod q$, then finding out R equals to $\alpha^{C.A}$ are computationally indistinguishable. Thus under Diffie Hellman assumptions the proposed protocol is secure against passive attacks.

3.3 Performance Evaluation

In this Section, performance of Tseng's modified protocol in the view of message size and computational complexity is analyzed. Moreover, a comparison of Biswas, Tseng and the modified protocol is prepared and it is made that the computational cost is decreased. To this aim, some notations are considered to measure the computational cost conveniently.

- $|m|$: Length of the message in bits;
- **TEXP**: represents the execution time for a modular exponentiation;
- **TINV**: indicates the execution time needed for a modular inverse;

- **TMUL**: shows Execution time of a modular multiplication in Biswas's Protocol;
- **TH**: represents the execution time required for a one-way hash function;
- **TXOR**: represents the execution time of the exclusive OR operation;

According to the discussion in Section 3.1, when the controller constructs a shared key in step one of the protocol, it should compute a public key which takes about $TEXP$ and K_i for all participants that is about $(n - 1)TEXP$. Thus, the computational complexity in step one for the controller is totally $n \cdot TEXP$. In addition, the controller needs $TEXP$ to calculate Y in step 2 and $(n - 1)TXOR + TH$ to compute the amount of Y_i for each participants and the common secret key. Finally, in case of controller the computational complexity in both steps together is $(n + 1)TEXP(n - 1)TXOR + TH$.

Obviously, there is less computational cost for other participants. Since in modified protocol the participants require to compute a Diffie Hellman shared key in step 1 in $2TEXP$; also in step 2 the computational complexity for finding amount of Y and group key is $TXOR + TH$. Thus, it can be concluded that in case of other participants the computational cost is $2TEXP + TXOR + TH$. The results for three protocols are prepared in Table 1. It can be easily understood that the Tseng's modified protocol achieves better result than others do while it is contributory protocol.

Thus, the modified protocol is suited for low-power participant as it takes less time to establish a group key or it can to be used in context of wireless sensor networks. The results of modified protocol are satisfactory due to the needs of ad hoc network that is dynamic and participants are not always connected to network.

In the view of the message size, when the participant nodes P_i ($1 \leq i \leq n-1$) want to create a shared key $K_i = \alpha^{e_i} \bmod q$ and send $\alpha^{e_i} \bmod q$ to controller, the message size for each participants is $|q|$. On the other hand, the generated common secret key $K = H(Y_i \oplus K_i, Y_1, Y_2, Y_3 \dots, Y_{n-1})$ should be broadcast by controller so the message size is $(n-1)|q|$.

Considering theoretical results, Table 1 demonstrates that Tseng's modified protocol achieves better results than other protocols.

Moreover, for controller that deals with more battery consumptions it is more important to reduce the cost of complexity. While for exponential modular calculation is almost same as inverse modular computation, both use the same procedure to find the output. Thus, $TEXP$ is nearly equal to $TINV$ in this case. It is remarkable that $TEXP$ and $TINV$ can be a degree one polynomial of $TXOR$. In view of controller, by considering $TINV$ equal to $TEXP$ and neglecting $TXOR$, proposed Tseng's modified protocol performs with approximately around 3 times less computational cost than Tseng and Biswas's protocol when the number of nodes increasing. The following analysis demonstrates it in details.

$$\frac{T_{Tseng\ protocol}}{T_{Tseng\ modified\ protocol}} = \frac{2nTEXP + (n-1)TINV + n \cdot TH}{(n+1)TEXP + (n-1)TXOR + n \cdot TH}$$

$$\frac{T_{Biswas\ protocol}}{T_{Tseng\ modified\ protocol}} = \frac{2nTEXP + (2n-5)TMUL + n \cdot TH}{(n+1)TEXP + (n-1)TXOR + n \cdot TH}$$

Table 1: Comparison between Biswas Protocol, Tseng Protocol and Tseng's Modified Protocol

	Biswas's [4] Group Key Exchange Protocol	Tseng [5] Group Key Exchange Protocol	Tseng's Modified Group Key Exchange Protocol
Contributiveness	No (Section 2.2)	YES	YES
Number of unicasting	$2n-2$	$n-1$	$n-1$
Number of broadcasting	1	2	2
Unicasting message size by each participant	$ q $	$ q $	$ q $
Broadcasting message size by each participant	0	0	0
Unicasting message size by controller	$ q $	0	0
Broadcasting message size by controller	$(n-1) q $	$(n) q $	$(n) q $
Computational costs for each participant	$3TEXP$	$3TEXP + n \times TH$	$2TEXP + TXOR + n \times TH$
Computational costs for controller	$2nTEXP + (2n-5)TMUL$	$2nTEXP + (n-1)TINV + n \times TH$	$(n+1)TEXP + (n-1)TXOR + n \times TH$

Considering $TEXP = 1$, $TINV \cong TEXP$, $TXOR \ll TMUL \ll TEXP$, $TH \ll TEXP$ based on the (15), (16) and (17) in section 4.2.

Moreover, the amounts of ε_1 , ε_2 , and ε_3 refers to small amount as bellow.

$$\varepsilon_1 = (n - 1)TXOR + n \cdot TH, \varepsilon_2 = n \cdot TH, \varepsilon_3 = (2n - 5)TMUL$$

$$\frac{T_{Tseng\ protocol}}{T_{Tseng\ modified\ protocol}} = \frac{2n + (n - 1) + \varepsilon_2}{(n + 1) + \varepsilon_1} = \frac{3n - 1 + \varepsilon_2}{n + 1 + \varepsilon_1} \quad (7)$$

$$\frac{T_{Biswas\ protocol}}{T_{Tseng\ modified\ protocol}} = \frac{2n + \varepsilon_3}{(n + 1) + \varepsilon_1} = \frac{2n + \varepsilon_3}{n + 1 + \varepsilon_1} \quad (8)$$

while n is increasing then,

$$\lim_{n \rightarrow \infty} \frac{T_{Tseng\ protocol}}{T_{Tseng\ modified\ protocol}} = \lim_{n \rightarrow \infty} \frac{3n - 1 + \varepsilon_2}{n + 1 + \varepsilon_1} = 3 \quad (9)$$

$$\lim_{n \rightarrow \infty} \frac{T_{Biswas\ protocol}}{T_{Tseng\ modified\ protocol}} = \lim_{n \rightarrow \infty} \frac{2n + \varepsilon_3}{n + 1 + \varepsilon_1} = 2 \quad (10)$$

Furthermore, Figure 2 illustrates computational cost growth based on two equations in (7), while execution time for Tseng protocol is about $3n - 1$ and for Tseng modified protocol is around $n + 1$, where n is number of nodes. Due to definition of group key exchange in Section 2.1.2, number of participants is more than two. The figure represents that there is a substantial growth for Tseng's protocol, while Tseng's modified protocol is going up gradually as the numbers of participants are increasing. It is anticipated that growth rate of computational cost for Tseng's protocol is much greater than Tseng modified protocol for more number of nodes.

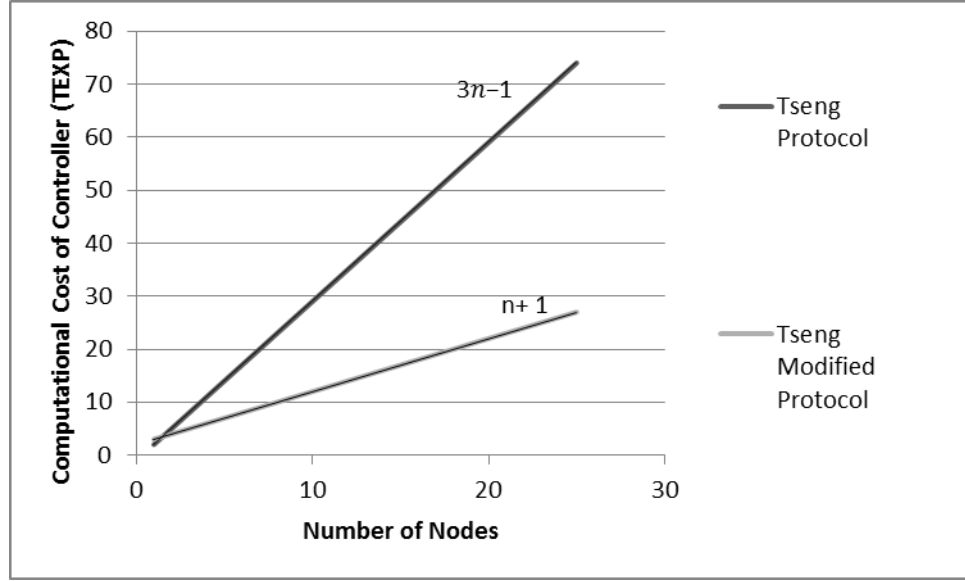


Figure 2: Computational Cost of Controller for 25 Nodes

In Table 1, by considering computational cost for each participant, while TXOR is negligible in comparison to TEXP, computational complexity is decreased around 1.5 times in proposed protocol as bellows.

$$\frac{T_{Tseng\ protocol}}{T_{Tseng\ modified\ protocol}} = \frac{3TEXP + n \cdot TH}{2TEXP + TXOR + n \cdot TH} = \frac{3 + \varepsilon_2}{2 + \varepsilon_1} \cong 1.5 \quad (11)$$

$$\frac{T_{Biswas\ protocol}}{T_{Tseng\ modified\ protocol}} = \frac{3TEXP}{2TEXP + TXOR + n \cdot TH} = \frac{3}{2 + \varepsilon_1} \cong 1.5 \quad (12)$$

For great number of nodes in (13) and (14), it seems that Tseng's protocol and Biswas protocol are going to be the same as Tseng's modified protocol. Biswas protocol in (14) acts better than Tseng's modified protocol, but it is noticeable that Biswas protocol is not a contributory protocol. While TXOR is very small amount in comparison to TEXP, it can be neglected. The amount of $\varepsilon = TH$ is considered as this small amount is multiple n times.

$$\lim_{n \rightarrow \infty} \frac{T_{Tseng\ protocol}}{T_{Tseng\ modified\ protocol}} = \lim_{n \rightarrow \infty} \frac{3T_{EXP} + n \cdot T_H}{2T_{EXP} + T_{XOR} + n \cdot T_H} = \lim_{n \rightarrow \infty} \frac{3 + n \cdot \varepsilon}{2 + n \cdot \varepsilon}$$

$$= 1(13)$$

$$\lim_{n \rightarrow \infty} \frac{T_{Biswas\ protocol}}{T_{Tseng\ modified\ protocol}} = \lim_{n \rightarrow \infty} \frac{3T_{EXP}}{2T_{EXP} + T_{XOR} + n \cdot T_H} = \lim_{n \rightarrow \infty} \frac{3}{2 + n \cdot \varepsilon}$$

$$= 0(14)$$

In conclusion, it is clear that Tseng modified protocol results are satisfied in comparison with Biswas protocol and Tseng's protocol. In the view of controller, based on (9) and (10) the proposed Tseng modified protocol achieves at least 3 times and 2 times less computational cost than Tseng's protocol and Biswas's protocol respectively. Moreover, in (11) and (12) for each participant around 1.5 times the computational cost is decreased. While Tseng modified protocol is a contributory group key exchange protocol, the computational cost of it is less than Tseng and Biswas protocol.

Chapter 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this Chapter, the modular interface for the Tseng's modified protocol is given and the details of Tseng's modified protocol implementation are provided. While Tseng's protocol leads to failure due to the fact that inverse of the established Diffie Hellman does not exist in some situations that explained in section 3.1, Tseng's protocol is not practicable and it is not implemented. In the view of implementation, Figure 4 illustrates the process of establishing a common secret key. In the mentioned module, the group controller C and the participant A presents member nodes that are automatically connected to the defined ad hoc WLAN called DH_GKE by running the application and they are willing to generate a group key. As it is displayed in Figure 4, each node is capable of being either the controller or a participant node. In addition, the received message is switched to proper case due to the role of nodes, controller C or usual member A. It is noticeable; Tseng's modified protocol is implemented in order to improve the key exchange part for Enhanced Scatter_{Light} protocol [26] to provide security. While, in Enhanced Scatter_{Light} protocol, the group key is static and it is defined before, Tseng's modified protocol brings the possibility of establishing a dynamic group key to Enhanced Scatter_{Light} protocol.

The first Section of this chapter is dedicated to explanation about procedures, routines and message transmitting in order to implement the modified protocol.

Subsequently, the experimental results of the modified protocol are demonstrated in Section 4.2. Moreover, the related code is provided in Appendices.

4.1 Implementation

The first challenging parts for implementation is choosing a large prime number such as q , and also to compute a primitive root such as α , while working with big numbers is needed to provide the security.

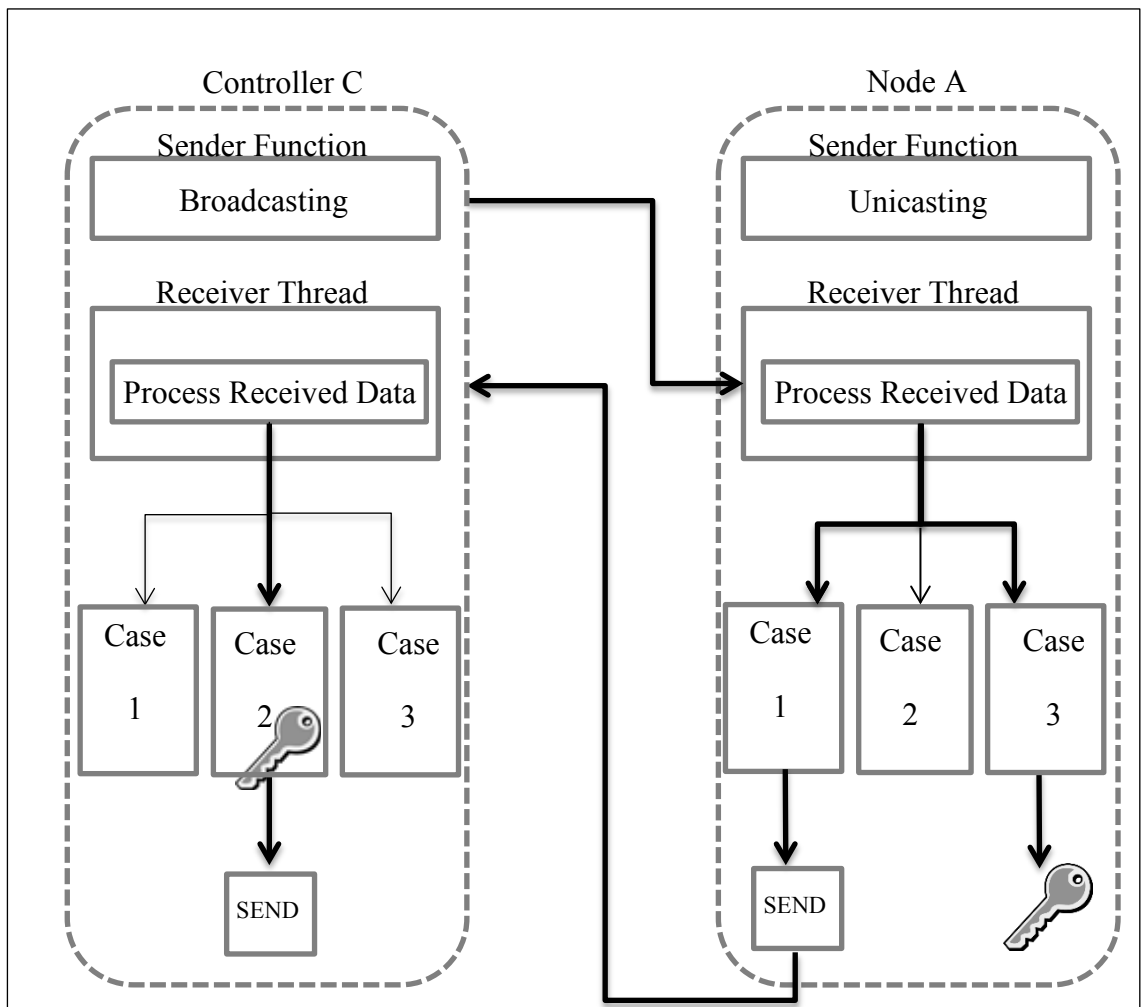


Figure 3: Modular Interface of Proposed Tseng Modified Protocol

When the ad hoc network is established and the initialization to define the proper q and α is done, the controller uses password of the user as e_c , described in Section 3.1, and broadcasts a message in network containing α^{e_c} . Meanwhile, the receiving

thread is always listening to the port. On receiving the message from controller, the participants should recognize appropriate case, whether the received message is a request to establish a key or contains the produced common secret key. According to the First Case, participants such as node A should count the shared key $K_i = \alpha^{e_c e_i} \bmod q$, and sends α^{e_i} to controller. Figure 5 shows the process of Case One that a participant node such as A wants to answer the controller request.

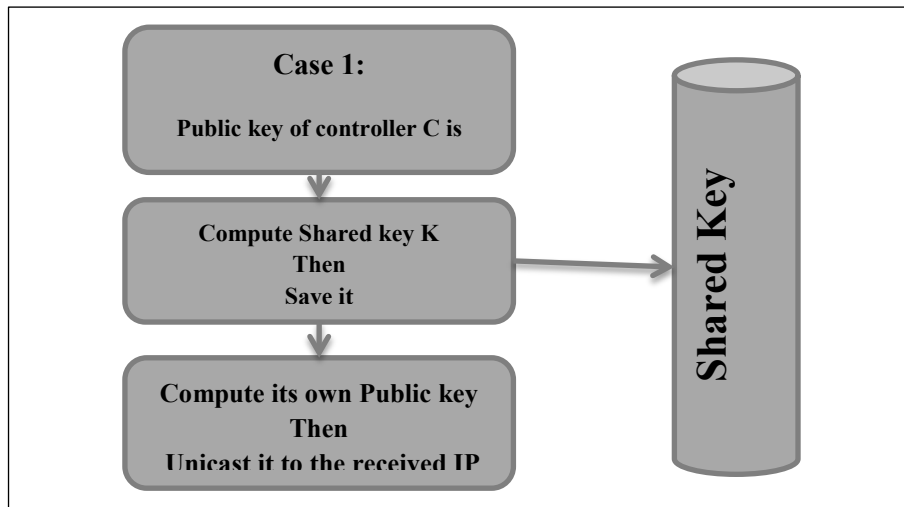


Figure 4: Flowchart of Case 1

In Case Two, the controller raises each received message from the participants to the private key e_c (the controller password), then chooses a random value to guess the amount of Y and calculate $Y_i = Y \oplus K_i$ for each participant. After that, the group controller saves Y_i in an array named Shared Key for each corresponding nodes. Finally, he/she prepares a message by use of Shared Key that is included Y_i and broadcast it in network. In addition, as the controller knows all parameter that is needed for common secret key, he/she computes the group key that is output of hash function. In this implementation, SHA-1 is considered as a one-way function. Figure 6 demonstrates how the shared key is produced in Case Two by controller.

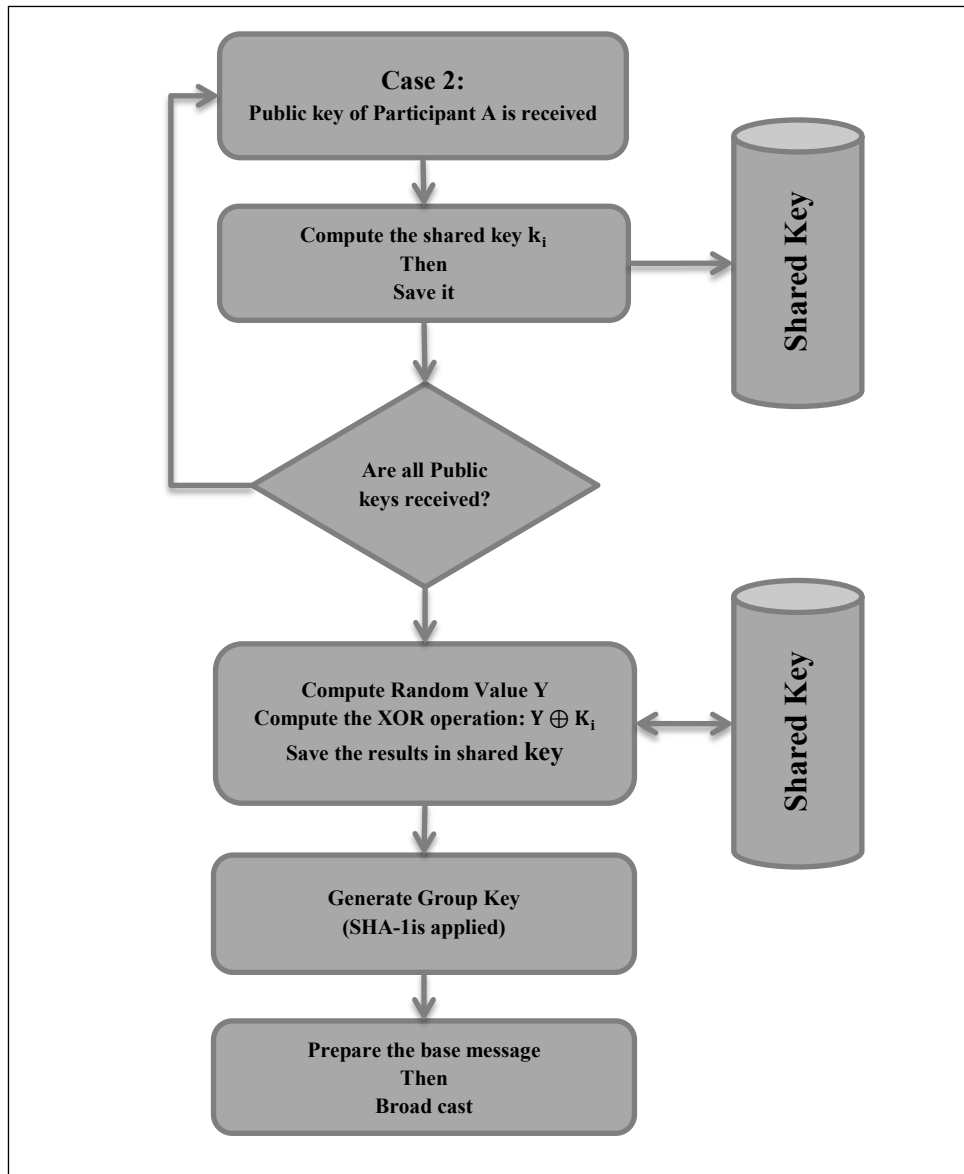


Figure 5: Flowchart of Case 2

Considering Case 1 and Case 2 are performed successfully, the participant nodes receive the final message from the controller that should be used to elicit the amount of Y . In Case 3, by using the Exclusive Or operation the derivation of Y from the message can be easily performed. After finding out the amount of Y , the participants apply it to the received message in order to feed the one-way function. In the end, the output of Sha-1 is a common secret key and the contributiveness group key exchanged protocol is finished. The flowchart of the Case 3 process is represented in

Figure 7. It is clear that the key together with encryption algorithms can provide secure communication in ad hoc network.

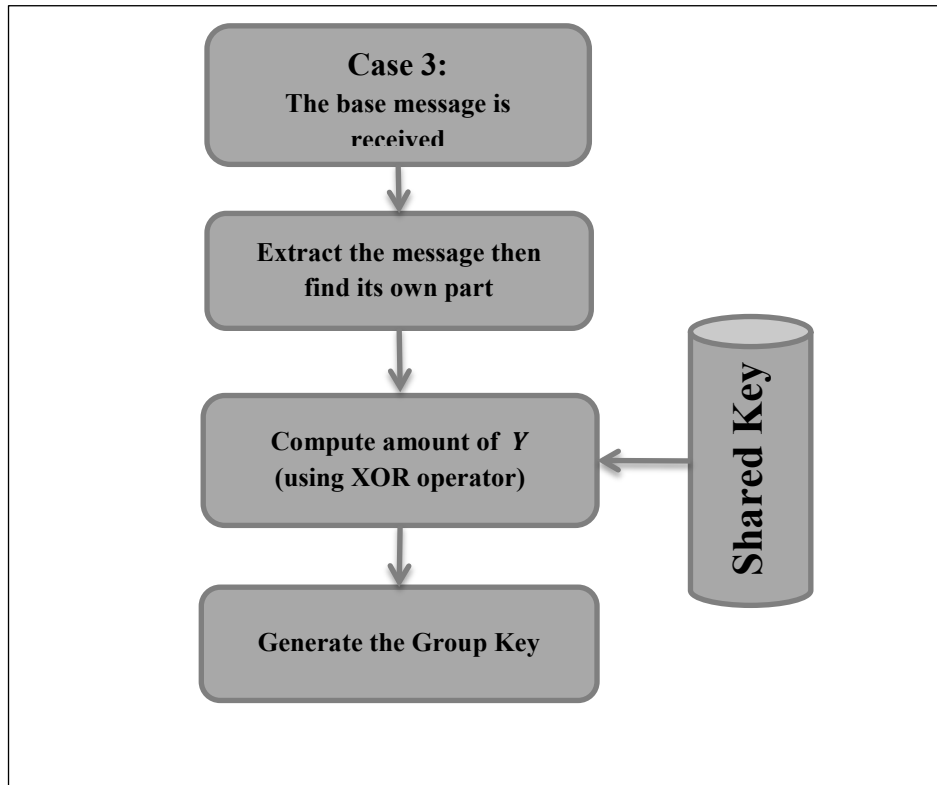


Figure 6: Flowchart of Case 3

4.2 Experimental Results and Settings

In this study, the experimental results that obtained after implementation and execution of modified protocol application are discussed. Visual C#.Net has been used as a programming language; the application is performed between three, four and five laptops, in distance of about 3 meters from each other's. The laptops that are used to construct the key contain Intel Core i5 CPU, 2GB RAM and Windows 7 as Operating System. Moreover, in order to dealing with big numbers computations, System.Numerics Namespace of .Net framework 4 is used [27]. It is noticeable that the firewall and Antiviruses should be turned off while the application is running.

For confident and less possible error, the average amount of total execution times is considered as an execution time of the Tseng modified protocol. To this aim, the application is run 20 times and the measured values of each runs are gathered in Table 3, and the average amount of all measured values is computed. Then, TEXP, TXOR and TH are measured and the computational cost of controller is calculated due to Table 1 in section 3.3 for Tseng modified protocol. Furthermore, computational cost and messaging cost of controller are retrieved by running the Tseng modified protocol and Table 4 is provided. Finally, the computational cost of controller achieved experimentally is compared to theoretical results in (15), (16) and (17).

Moreover, some parameters such as prime number range and the private value or password of each node should be set in the application. It is clear that the most important parameter, which is a prime number q , should be set the same for all participants based on Diffie Hellman Key Exchange Definition is explained in Section 2.1.3. Table 2 represents initialization to set up the application and get the results. In order to provide security, working with big numbers is needed thus the parameter q that is considered in this study is equal to a 100 digit prime number and the primitive root α that is equal to 50 digit number. Moreover, the amount of password x_A that is an exponent in Diffie Hellman method is considered as a 50 digit number.

Table 2 : Initialization of Parameters

Parameter	Size	Amount
prime number q	100 digit	207472224677348520782169 522210760858748099647472 111729275299258991219668 475054965831008441673255 0077
generator α	50 digit	464847298035401831018301 678756237887945334412167 79
Private amount x_A or password	50 digit	487050913552388827788429 092300567121408134601578 99

Base on Initialization parameters in Table 2, the execution time; TEXP, TXOR and TH; are measured as below.

$$TXOR = 0.001(ms), TH = 0.010(ms)$$

To achieve less possible error in computing TEXP, different ranges of numbers are considered. Table 3, shows TEXP for three categories; small numbers, medium numbers and big numbers. It is noticeable that small numbers are assumed as at most 33 digit numbers, medium numbers are at most 66 digit numbers and big numbers are 100 digit numbers.

Table 3: TEXP (in millisecond) For Three Different Categories

Base \ Exponent	small	medium	big
small	1	1.6	3.03
medium	1.02	2.08	3.23
big	1.4	2.19	4.01

Regarding to Table 3 the average amount for TEXP is computed as below.

$$TEXP = 2.17 (ms)$$

Furthermore, due to Tseng's modified protocol in section 3.3, computational complexity for controller is equal to $(n + 1)TEXP + (n - 1)TXOR + n \cdot TH$ theoretically, and it is computed for 3, 4 and 5 participants as below.

$$n = 3, \tag{15}$$

$$\begin{aligned} 4 \cdot TEXP + 2 \cdot TXOR + 3 \cdot TH &= 4 \times 2.17 + 2 \times 0.010 + 3 \times 0.001 \\ &= 8.703 (ms) \end{aligned}$$

$$n = 4, \tag{16}$$

$$\begin{aligned} 5 \cdot TEXP + 3 \cdot TXOR + 4 \cdot TH &= 5 \times 2.17 + 3 \times 0.010 + 4 \times 0.001 \\ &= 10.884(ms) \end{aligned}$$

$$n = 5, \tag{17}$$

$$6 \cdot TEXP + 4 \cdot TXOR + 5 \cdot TH = 6 \times 2.17 + 4 \times 0.010 + 5 \times 0.001) \\ = 13.065 (ms)$$

On the other hand, by running the application 20 times on 5 laptops, the computational cost and messaging cost for controller are measured. Then one laptops stops running the application and the results of 20 times running the application on 4 laptops are gathered, then for 3 laptops the same as previous one messaging cost and computational cost are measured. Finally, all results of running the application on 5, 4 and 3 laptops are provided and shown in Table 4. The table presents the average messaging cost, average execution time and the computational cost in millisecond after 20 runs. Execution time is referred to the time for executing the Tseng's modified protocol and it is measured when the protocol starts until the group key is generated. Messaging time means the time for sending and receiving messages in Tseng's modified protocol. As it is shown in Figure 3, messaging time includes the time after sending the message by controller until it is received by Process Received Data in controller part.

Computational cost, which is provided in row before the last row, is obtained by subtraction of the average execution time and average messaging time. As it is shown in Table 4, the computational cost of the controller that is achieved by experimental results is approximately equal to theoretical amounts in equations (15), (16) and (17).

Table 4: Average Messaging Time, Average Execution Time and Computational Cost (in millisecond) for 20 Runs of Tseng's Modified Protocol Based on Initialization Parameter in Table 2

Number of nodes	5		4		3	
Run \ Time	Execution time (ms)	Messaging Time (ms)	Execution time (ms)	Messaging Time (ms)	Execution time (ms)	Messaging Time (ms)
1	21.7	6.82	19	6.46	15.8	6.45
2	19.8	6.76	17.9	6.76	15.3	6.63
3	19.9	6.84	18.3	6.53	15	6.29
4	20.1	6.85	18.1	6.47	15.3	6.28
5	19.9	6.87	18	6.48	15.1	6.42
6	19.9	6.75	18.4	6.57	15.7	6.29
7	19.4	6.67	17.5	6.66	15.2	6.36
8	20.1	6.71	17.7	6.8	15.4	6.68
9	19.7	6.79	17.8	6.63	15.7	6.38
10	19.7	7.01	17.7	6.45	15.2	6.24
11	20.1	6.77	17.4	6.54	15.3	6.29
12	21.8	6.92	17.5	6.4	15.4	6.26
13	19.8	6.75	17.4	6.56	15.4	6.53
14	22.1	6.69	18.1	6.39	15.1	6.34
15	20.1	6.78	17.8	6.7	15.5	6.48
16	20.3	6.91	17.6	6.58	15.5	6.54
17	21	6.81	18	6.72	15.4	6.27
18	19.7	6.73	17.9	6.41	15.4	6.33
19	20.1	6.68	17.8	6.59	15.5	6.34
20	20.1	6.79	17.9	6.61	15.1	6.44

Average	20.265	6.795	17.89	6.5655	15.365	6.392
Computational Cost	13.47		11.32		8.973	
Computational Cost due to equation (15),(16),(17)	13.065		10.88		8.7	

Moreover, Figure 7 illustrates computational cost of controller in Tseng's modified protocol regarding to results in Table 4. As the figure shows, it seems that the computational cost growth is linear. That is similar to Figure 2.

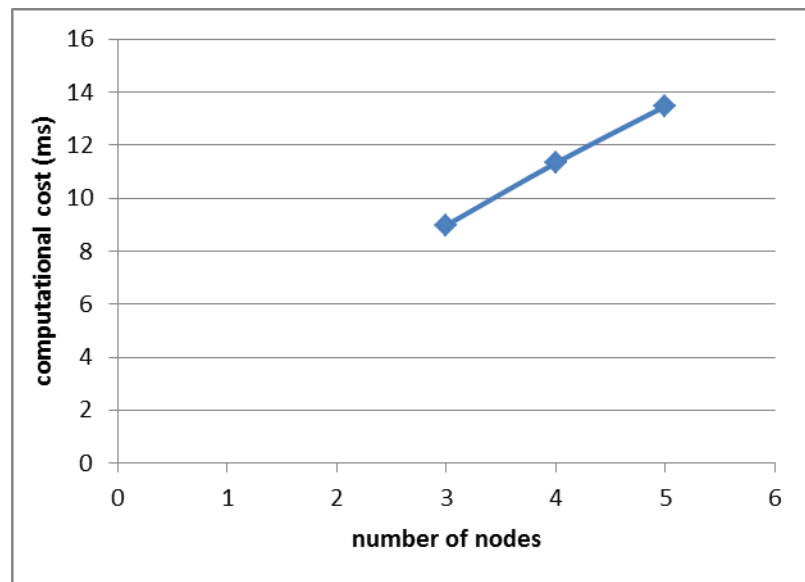


Figure 7 : Computational Cost Based On Experimental Result in Table 4 for Tseng's Modified Protocol

Chapter 5

CONCLUSION

During last decades, it is unavoidable to use cryptographic algorithms to provide security for WLAN or ad hoc network. Moreover, creating a secret key is one of the significant issues, which is needed for all cryptographic algorithms. Thus establishing a common shared key for all members of the ad hoc network or Wireless Sensor Networks (WSN) due to the properties of these types of network that participants are low power devices with not much memory is a matter of debate.

In this study, Tseng's modified protocol, which is modification of Tseng's protocol [5], is proposed and implemented. Although Tseng's group key exchange protocol provides contributiveness, the protocol cannot perform in some situations when the generated DH style key is not invertible and leads to failure. Thus, in the proposed protocol a modification is done on Tseng's protocol to cover these failures, reduce the computational complexity of the protocol, and obsoletely provide security to be well suited for small devices in WSN or ad hoc networks. Furthermore, theoretical results show that the proposed protocol reduces the computational complexity of controller and all contributors approximately around 3 times and 1.5 times in comparison to Tseng's protocol respectively. Accordingly, the experimental results gained for Tseng's modified protocol are complying.

REFERENCES

- [1] M. Hietalahti, "Key Establishment in ad hoc Networks," in *Seminar on Network Security, Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory(HUT TML)*, Helsinki, Finland, fall 2000.
- [2] "Diffie–Hellman key exchange," WIKIPEDIA, [Online]. Available: http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange#cite_note-1. [Accessed 3 December 2013].
- [3] K. Becker and U. Wille, "Communication Complexity of Group Key Distribution," in *5th ACM Conference on Computer and Communications Security*, California, USA, November 1998.
- [4] G. Biswas, *The Institution of Engineering and Technology(IET) Information Security*, vol. 2, no. 1, p. 12–18, March 2008.
- [5] Y.-M. TSENG and T.-Y. WU, "Analysis and Improvement on a Contributory," *INFORMATICA International Journal*, vol. 21, no. 2, p. 247–258, April 2010.
- [6] D. Boneh, "The Decision Diffie-Hellman problem," *Algorithmic Number Theory, Third International Symposium, ANTS-III*, vol. 1423, pp. 48-63, 21–25 June 1998.
- [7] M. "WikiHow," [Online]. Available: <http://www.wikihow.com/Break-WEP->

Encryption. [Accessed 03 December 2013].

- [8] C. K. Toh, *ad hoc Wireless Networks: Protocols and Systems*, Prentice Hall PTR Upper Saddle River, NJ, USA ©2001, December 3rd 2001.
- [9] Y.-M. Tseng, "An Efficient Two-Party Identity-Based Key Exchange Protocol," *Informatica IOS Press, ISSN:0868-4952*, vol. 18, no. 1, pp. 125-136, January 2007.
- [10] J. Nam, K. Lee, J. Paik, . W. Paik and D. Won, "Security Improvement on a Group Key Exchange Protocol for Mobile Networks," in *Computational Science and Its Applications (ICCSA)*, Santander, Spain, June 20-23, 2011.
- [11] M. Manulis, "Contributory group key agreement protocols, revisited for mobile ad hoc groups," in *IEEE International Conference on Mobile Ad hoc and Sensor Systems Conference*, Washington DC, USA, 7-7 Nov, 2005.
- [12] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE TRANSACTIONS ON INFORMATION THEORY*, Vols. IT-22, no. 6, pp. 644-654, November 1976.
- [13] "Computational Diffie–Hellman assumption," WIKIPEDIA, [Online]. Available: http://en.wikipedia.org/wiki/Computational_Diffie%E2%80%93Hellman_assumption. [Accessed 15 December 2013].

- [14] "Decisional Diffie–Hellman assumption," WIKIPEDIA, [Online]. Available:
http://en.wikipedia.org/wiki/Decisional_Diffie%E2%80%93Hellman_assumption.
[Accessed 15 December 2013].
- [15] "Passive Attack," WIKIPEDIA, [Online]. Available:
http://en.wikipedia.org/wiki/Passive_attack. [Accessed 3 December 2013].
- [16] I. Ingemarsson, D. Tang and C. Wong, "A conference key distribution system," *IEEE Information Theory Society*, vol. 28, no. 5, pp. 714 - 720, September 1982.
- [17] M. Steiner, G. Tsudik and M. Waidner, "Diffie-hellman Key Distribution Extended to Group Communication," in *3rd ACM Conference on Computer and*, New Delhi, India, March 1996.
- [18] Y. Kim, A. Perrig and G. Tsudik, "Tree-based group key," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 1, pp. 60-96, February 2004.
- [19] Y. Kim, A. Perrig and G. Tsudik, "Group Key Agreement Efficient in Communication," *IEEE Transactions on Computers*, vol. 53, no. 7, pp. 905-921, July 2004.
- [20] T. Koshy, "Multiplicative Functions," in *Elementary Number Theory with Applications, Second Edition*, California, USA, Elsevier, May 8, 2007, pp. 355-412.

- [21] B. Schneier, "Modular Exponentiation," in *Applied Cryptography, Second Edition*, New York, USA, Wiley, 1996, pp. 244-275.
- [22] "Secure Hash Standard (SHS)," NIST/NSA, Federal Information Processing Standards Publication (FIPS) 180-2, Gaithersburg, MD, USA, 2005.
- [23] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *1st ACM conference on Computer and communications security*, Fairfax, VA, USA, 03 - 05 November 1993.
- [24] W. Stallings, CRYPTOGRAPHY AND NETWORK SECURITY PRINCIPLES AND PRACTICE, fifth Edition, New York, USA: Pearson, 14 January 2010.
- [25] R. . F. Churchhouse, "Modular Addition and Subtraction of Letters," in *Codes and ciphers Julius Caesar, the Enigma and the internet*, New York, USA, THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE, 2004, pp. 11-68.
- [26] A. Chefranov, S. M. Alavi Abhari, H. Alavizadeh and M. Farajzadeh zanjani, "Secure True Random Number Generator in WLAN/LAN," in *ACM Digital Library*, Aksaray, Turkey, 2013.
- [27] "BigInteger Structure," Microsoft Developer Network, [Online]. Available: [http://msdn.microsoft.com/en-us/library/system.numerics.biginteger\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.numerics.biginteger(v=vs.110).aspx). [Accessed 06 October 2013].

APPENDICES


```

p1.Start();
}

private void timer1_Tick_1(object sender, EventArgs e)
{
    IPEndPoint myHostInfo = Dns.Resolve(Dns.GetHostName());
    IP = myHostInfo.AddressList[0].ToString();
    lblIP.Text = IP;

    net_view();
}
void net_view()
{
    nb = new NetworkBrowser();
    lstNetworks.Items.Clear();
    string nbh = "";
    foreach (string pc in nb.getNetworkComputers())
    {
        try
        {
            addresslist = Dns.GetHostAddresses(pc);
            foreach (IPAddress address in addresslist)
            {
                nbh = address.ToString();
            }
        }
        ListViewItem item = new ListViewItem(pc);
        item.SubItems.Add(nbh);
        lstNetworks.Items.Add(item);
        nbh = "";
    }
    cnt_nodes = lstNetworks.Items.Count;
    lbl_nodes.Text=cnt_nodes.ToString();
}

```

Immediately after establishing the ad hoc network, a thread named P1 starts working. P1 is responsible of listening to port for any received message. A procedure named Process_Received_Data is called by P1. The details and codes of the related procedure are explained in computation process part.

```

void receiver()
{
    while (true)
    {
        receive_byte_array = listener.Receive(ref groupEP);
        received_data = Encoding.ASCII.GetString(receive_byte_array, 0,
        receive_byte_array.Length);
        data = new byte[receive_byte_array.Length];
    }
}

```



```

data = Encoding.ASCII.GetBytes(received_data);
Process_Recived_Data(data);
}
}

```

One of the most important part of initialization is setting a prime number q and computing a proper amount α for q . Furthermore, to achieve more security it is assumed that q should be greater than 5000 and α should be greater than half of q . When the button 1 is clicked, q and α are computed and the calculated amounts are shown in linked text boxes. The codes that generate q and α are as bellows.

```

private void button1_Click(object sender, EventArgs e)
{
    Boolean prime = false;
    prime_q = Convert.ToInt64(primeX.Text);
    while (!prime)
    {
        prime = true;
        for (int j = 2; j < System.Math.Sqrt(prime_q)+1 ; j++)
        {
            if ((prime_q % j) == 0) { prime = false; break; }
        }
        if (prime) txt_qFind.Text=prime_q.ToString();
        else prime_q++;
    }
    //-----
    // finding proper Alpha
    //-----
    Boolean[] prime_root_array = new Boolean[prime_q]; // max = 7000
    BigInteger temp = 0;
    Boolean RT=false;
    for (long k1 = prime_q/2; k1 < prime_q - 1; k1++)
    {
        //count = 0;
        RT = true;
        for (int r = 0; r < prime_q; r++) prime_root_array[r] = false;
        for (int k2 = 1; k2 <= prime_q - 1; k2++)
        {
            temp = BigInteger.ModPow(k1, k2, prime_q);
            //-----//
            if (prime_root_array[(int)temp] == false)
                prime_root_array[(int)temp] = true;
            else
            {
                RT = false;
                break;
            }
        }
    }
}

```

```

}
if (RT == true)
{
Alpha = k1;
txt_Alpha.Text=Alpha.ToString(); break;
}
}
Shared_Key = new string[1stNetworks.Items.Count + 5];
Private_No = Convert.ToInt32(txt_password.Text);
pointer_arrey = 0;
Txt_SessionKey.Text = "";
}

```

It is also remarkable that user should enter a password as a private number in related text box.

A.2 Computation Process

When the initialization is completely done, it will be possible for each member of the group to request the group key. While the button “Generate Group Key” is clicked a request message of first type, broadcast to networks. Other nodes received the message switch to case 1, calculate DH shared key and prepare a message of second type and send it back to requester (controller). Then the controller randomly chooses an amount between 5000 and 2,000,000,000 after calculating the shared key and generating group key again broadcast a message. Other nodes that received the message of third type, switch to case three to retrieve the amount of Y and generate the shared key. The procedure of this process named “Process_received_data” and is called by a receiver thread.

```

private void btn_sharedKey_Click(object sender, EventArgs e)
{
stopWatch.Start();

```

```

num = 0;
num = RandomNumber(50000, 2000000000);
//-----
Private_No = Convert.ToInt32(txt_password.Text);
sending("1," + BigInteger.ModPow(Alpha, Private_No, prime_q).ToString(),
"192.168.255.255"); //broadcasting
}

```

The codes related to receiver thread are as bellows.

```

void receiver()
{
while (true)
{
receive_byte_array = listener.Receive(ref groupEP);
received_data = Encoding.ASCII.GetString(receive_byte_array, 0,
receive_byte_array.Length);
data = new byte[receive_byte_array.Length];
data = Encoding.ASCII.GetBytes(received_data);
Process_Recived_Data(data);
}
}

private void Process_Recived_Data(Byte[] data)
{
if (!(groupEP.Address.ToString() == lblIP.Text))
{
switch (received_data.Split(',')[0].ToString())
{
case "1":
{
Shared_Key[0] =
BigInteger.ModPow(BigInteger.Parse(received_data.Split(',')[1].ToString()),
Private_No, prime_q).ToString() + "," + groupEP.Address.ToString();
sending("ACK," + BigInteger.ModPow(Alpha, Private_No, prime_q),
groupEP.Address.ToString());
break;
}
case "2":
{
y = BigInteger.ModPow(Alpha,num,prime_q);

Shared_Key[pointer_arrey] = (y ^
BigInteger.ModPow(BigInteger.Parse(received_data.Split(',')[1].ToString()),
Private_No, prime_q)).ToString();
Shared_Key[pointer_arrey] = Shared_Key[pointer_arrey] + "," +
groupEP.Address.ToString();
if (pointer_arrey == (cnt_nodes-1))
{
//----- generating key and broadcasting
string key = "";
for (int i = 1; i < (cnt_nodes-1); i++)
{

```

```

}
key += Shared_Key[cnt_nodes-1];
sending("3," + key, "192.168.255.255");
key = y.ToString() + "," + key + ",";
//-----generate session key:
SHA1 sha = new SHA1CryptoServiceProvider();
byte[] data_sha = new byte[key.Length];
data_sha = Encoding.ASCII.GetBytes(key);
SessionKey = Encoding.ASCII.GetString(sha.ComputeHash(data_sha));
text_add(SessionKey);
}
break;
}
case "3":
{
BigInteger y=0;
string[] parts;
string key="";
parts = received_data.Split(',');
int cnt = 0;
foreach (string p in parts)
{
if (cnt != 0)
key += p + ",";
if (p == lblIP.Text)
{
y = BigInteger.Parse(parts[--cnt].ToString());
y = y ^ BigInteger.Parse(Shared_Key[0].Split(',')[0].ToString());
}
cnt = cnt + 1;
}
key = y.ToString() + "," + key;
SHA1 sha = new SHA1CryptoServiceProvider();
byte[] data_sha = new byte[key.Length];
data_sha = Encoding.ASCII.GetBytes(key);
SessionKey = Encoding.ASCII.GetString(sha.ComputeHash(data_sha));
text_add(SessionKey);
break;
}
}
}

private int RandomNumber(int min, int max)
{
Random random = new Random();
return random.Next(min, max);
}

```

In order to estimate an execution time of the protocol the stopwatch is considered that is started in the beginning of protocol exactly when the user clicks “Generate Group Key” button. Moreover, send and receive data User datagram protocol (UDP) is used. Following codes are related to UDP.

```

private void sending(string send_buffer, string ip_adr)
{
try
{
sending_end_point = new IPEndPoint(IPAddress.Parse(ip_adr), 11000);
sending_socket.SendTo(Encoding.ASCII.GetBytes(send_buffer.ToString()),
sending_end_point);
}
catch (Exception send_exception)
{
Console.WriteLine(" Exception {0}", send_exception.Message);
}
}
}

```

A.3 Printing the Results

In this step, while all three cases are done successfully, the generated group key is presented in a key text box and the stopwatch that was started upon the request is broadcast, is stopped. Furthermore, it is possible for each participant to change its password and request another group key. The codes related to print the key and execution time are as follows.

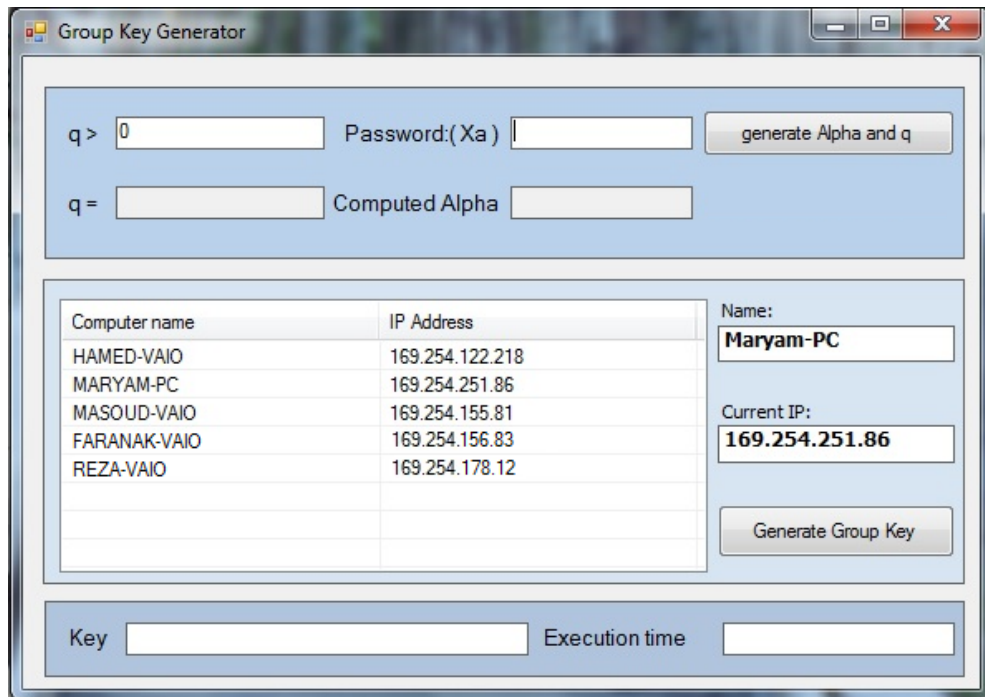
```

private void text_add(string str)
{
if (Txt_SessionKey.InvokeRequired == true)
{
setDisplay d = new setDisplay(text_add);
this.Invoke(d, new object[] { str });
}
else
{
Txt_SessionKey.ForeColor = System.Drawing.Color.Blue;
Txt_SessionKey.Text = str;
stopWatch.Stop();
TimeSpan ts = stopWatch.Elapsed;
string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
ts.Hours, ts.Minutes, ts.Seconds,
ts.Milliseconds / 10);
Txt_exeTime.Text = "RunTime " + elapsedTime;
}
}
}

```

APPENDIX B: User Guide

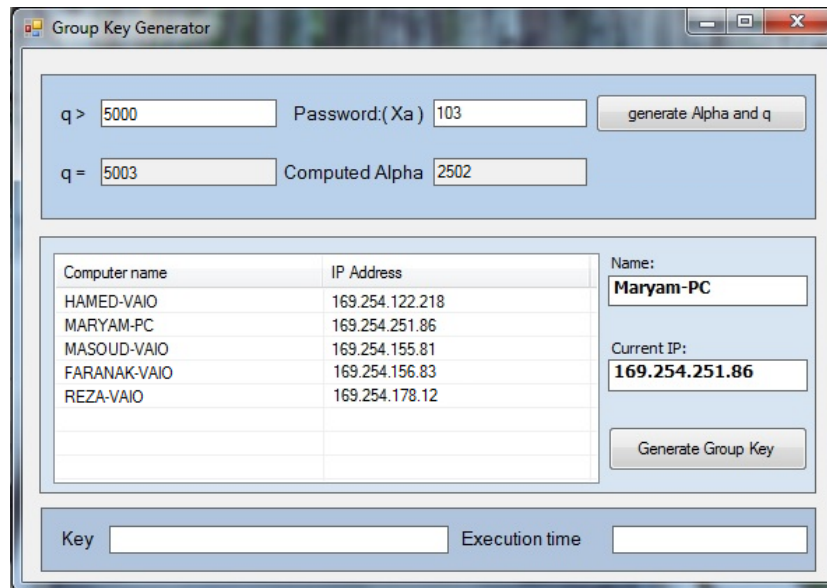
By running the application on a laptop, it joins to ad hoc network and user can see the IPs and laptop's names of other participants in a list. Moreover, user's IP address and laptop's name are shown in a text box.



The screenshot shows the 'Group Key Generator' application window. It features an initialization panel at the top with input fields for 'q >' (containing '0') and 'Password:(Xa)'. A 'generate Alpha and q' button is located to the right. Below this, there are output fields for 'q =' and 'Computed Alpha'. The main area contains a table of computer names and IP addresses, a 'Name:' field with the value 'Maryam-PC', and a 'Current IP:' field with the value '169.254.251.86'. A 'Generate Group Key' button is positioned below the table. At the bottom, there are fields for 'Key' and 'Execution time'.

Computer name	IP Address
HAMED-VAIO	169.254.122.218
MARYAM-PC	169.254.251.86
MASOUD-VAIO	169.254.155.81
FARANAK-VAIO	169.254.156.83
REZA-VAIO	169.254.178.12

Before demanding to generate a group key, some information should be set. There is an initialization panel in top of the application, which asks about amount of q and user password. When the informations are entered, user can click on the button named “Generate Alpha and q ” to compute the proper amounts. Related text boxes are filled with proper amounts when the computation is complete. Then user is able to request a group key as many times as he/she wants.



Then user click on “Generate Group Key” button and the established common secret key between participant nodes will be presented in last text boxes, also the execution time will be calculated and shown.

