AN ABSTRACT OF THE DISSERTATION OF

Özkan M. Erdem for the degree of

Doctor of Philosophy in

Electrical and Computer Engineering presented

on March 18, 2005.

Title: Efficient Key Management Protocols for Mobile Ad Hoc Networks

Abstract approved: Redacted for Privacy

In this thesis, novel solutions are proposed for key management issues in mobile ad hoc networks.

Presented Hierarchical Binary Tree (HBT) based model is distributed, self-organizing, scalable and does not employ online key distribution authority or group manager. Two different group authentication and group key establishment protocols are proposed for the users who form an ad hoc group with distributed trust model. Initially proposed protocols are based on public key cryptography and do not use specific algorithm. However, members can establish the keys faster with proposed customized hybrid scheme which combines elliptic curve cryptography, modular squaring operations and secret key encryption algorithm. Proposed HBT based model provides complete backward and forward security in case of modification in membership and it has comparable efficiency to the other HBT based schemes which employ real time key distribution authority.

Mutual authentication and link encryption can be achieved in wireless sensor network only with public key cryptography if there are no pre-distributed keys. However, constraints in resources make fully public key operations not affordable on sensor. Three different authenticated key establishment protocols are proposed with an objective of being respectful to constraints. Sensor needs to make only modular or cyclic convolution multiplications, and expensive public key decryption operation is executed at the data processing station side. Schemes require small size of code and achieve the least sensor processing time in comparison with fully public key cryptography based protocols.

Efficient Key Management Protocols for Mobile Ad Hoc Networks

by
Özkan M. Erdem

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented March 18, 2005
Commencement June 2005

Doctor of Philosophy dissertation of Özkan M. Erdem presented on March 18, 2005.

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical and Computer Engineering

Redacted for Privacy

Associate Director, School of Electrical Engineering and Computer Science

Redacted for Privacy

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Redacted for Privacy

Özkan M. Erdem, Author

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation and gratitude to Prof. Bella Bose, my thesis advisor, for his guidance and encouragement. I have been fortunate in being able to benefit from his experiences.

I am also thankful to my program committee members and to Dr. Erdal Paksoy at Texas Instruments, Inc. for their valuable comments and support in conducting this research and producing the papers which formed a basis for this thesis.

I am especially grateful to my wife, Marina, for her love and patience. She provided all of the emotional support to help me when things seemed difficult. Also I thank my parents for their encouragement, support and caring. Finally, I would like to thank my daughter, Melissa, for cheerful moments during hard times of my research.

Özkan M. Erdem

Dallas, TEXAS

March 2005

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

To my wife and my daughter,

EFFICIENT KEY MANAGEMENT PROTOCOLS FOR MOBILE AD HOC NETWORKS

CHAPTER 1

GENERAL INTRODUCTION

The term of mobile ad hoc networking refers to technology which provides communication between two or more mobile computers using standard network protocols in wireless manner and free of constant network topology [6]. Devices move and organize themselves arbitrarily. Networking functions are performed by the devices in a self-organizing manner. They can communicate with all of the other wireless enabled devices either directly or using hopping techniques. Dynamically relocating nodes change the topology of the network rapidly.

Due to mobility of devices, traditional security models designed for fixed-network topologies may not be fully applicable in ad-hoc networks [57]. Therefore, securing mobile ad hoc networks has brought new challenges and opportunities in terms of trust model, key management, and the standard security services such as confidentiality, authentication and integrity.

Basically two trust models can be introduced to mobile ad hoc networks:

1. *Centralized key management and trust model*: In this model, allocated base station or access point act as trusted key distribution authority and provides security related services for mobile ad hoc network.
2. *Distributed key management and trust model*: This model does not enforce any centralized key management over the network devices. Network comprises of self-organizing devices which have no access to any online

central resource, server or to internet. Typical example of this type of network is a setup of secure business meeting with the devices which have no access to company's centralized security services.

Chapter 1.1 and Chapter 1.2 give more details on mentioned trust models and address the common issues and open problems. Contributions done to the security of mobile ad hoc networking are given in Chapter 1.3.

1.1 Key Management in Centralized Trust Model

In centralized trust model, central authority establishes the trust among devices and manages the group keys [14, 39, 42 and 46]. Kerberos authentication and key management service is a well-known example of this model [14]. Kerberos key distribution center (KDC) is an on-line available server which authenticates the mobile device, accepts to the network and grants tickets to use the services of network. However, in Kerberos authentication protocol previously established secret key between user and KDC is password protected in user side and password can be easily guessed with appropriately performed dictionary attacks [52]. Besides, mutual authentication between user and KDC cannot be provided by only secret-key cryptographic algorithm. Many recent network authentication protocols support public key cryptography which offers robust solutions for mutual authentication between client and servers [52]. However, public key crypto operations are relatively more expensive than secret-key operations and consume more computational and bandwidth resources. Due to computational, communication and memory constraints of battery-powered mobile devices, it is crucial to find effective usage of public key

cryptographic methods to provide key freshness and mutual authentication features to Kerberos.

Mobile ad hoc sensor network is a good example for mobile ad hoc networks which has centralized trust model. Sensors are battery-powered computationally weak devices and functionality of sensors is delivering the sensed data to the collection stations. Sensor network consists of one central base station and hundreds of tiny sensors. Operation control, surveillance, vehicle tracking, environment monitoring, health care and maintenance of manufacturing systems are some of the applications which can use sensor networks. The privacy and security in sensor networks become essential when the security of deployment area cannot be provided. Encryption link key establishment and mutual authentication between data collection station and sensor are major building blocks of sensor network security. As alternative to using fully secret-key cryptography, public key cryptography based schemes offer self-organized trust establishment with the help of key certificates. Moreover, compromise of any device does not pose a security risk to other sensors. However, sensors cannot afford expensive public key operations with their extremely low computational power and small-size memory. Therefore, very light-weight public-key based mutual authentication and link key establishment protocols should be designed and carefully adapted to the sensor network environment.

1.2 Key Management in Distributed Trust Model

In distributed trust model, security does not rely on any trusted authority or fixed central server. Trust is totally decentralized, distributed among entities, and devices establish self-organized trust relations [6, 7, 13, 21 and 57]. Model assumes that trust relationships are unidirectional and taking place between two entities.

Key information can be distributed among devices prior to forming the group and devices store link key tables. Each key in the table is used to encrypt the traffic between only two devices and every table essentially has N-1 distinct pairwise secret keys, where N shows the total number of devices. Although this scheme is especially suitable for computationally weak devices, there is a great effort in pre-distribution of the pair-wise keys when there is no prior trust relationship. Besides, each device should carry all the pair-wise keys for potential future communications. This significantly decreases the memory and bandwidth efficiency of the scheme when the number of interacted devices increases. To overcome this inefficiency, all pair-wise link keys can be derived from one *group master secret key* which is stored in every device [4]. However, in this method compromise of any device can also compromise the whole network since the master secret key is revealed.

A good key management policy is crucial factor in establishment of trust model. In both trust models group key management includes several procedures for establishment and maintenance of group keys. Maintenance procedure contains group key renewal due to modifications in group membership or due to expiration of the group key. In distributed trust model group key establishment is also distributed. Two possible group key establishment models are applicable to distributed trust model. In first approach all group members can contribute for the group key generation. Alternatively, in partially contribution approach selected group member(s) establish the group keys and securely share with other members.

The use of public key techniques along with the key certificates can help to overcome the difficulty of node-to-node key establishment. There are some recent proposals based on modular exponentiation techniques for the management of encryption keys. However, using modular exponential techniques may not be possible for the network

formed by resource constrained portable devices. Hierarchical binary tree (HBT) is one of the previously studied techniques to manage the keys in distributed group structure [7, 11, 12, 35 and 46]. Most of the relevant proposals involve a key distribution authority which maintains a tree of keys and manages the group structure. Key distribution authority provides the group key to members and initiates the key update procedures by sending encrypted messages to members. However, in distributed trust model it is not possible to employ an online key distribution server. Devices should authenticate each other and establish the encryption link keys. A recent research in [7] proposes choosing one of the group members as a group manager and assigning all group key management functionalities to group manager. Note that this scheme brings a single point of attack and failure in the group since all keying material is managed centrally by the Group Manager.

As a conclusion, design of resource efficient group key management schemes stays as an open problem in distributed trust model when there is no online trusted key distribution authority.

1.3 Contributions

In Chapter 2, a practically secure protocol is proposed to build efficient hierarchical binary tree trust model for mobile ad hoc networks which has no trusted third party to maintain the keys. The proposal, which is called as EDKM protocol, employs a secure distributed model based on secret-key encryption algorithm and one-way hash function. The EDKM protocol also includes group keying and group membership change operations for single join/eviction and multiple joins/evictions. For a balanced tree associated to n users, each member holds $(2K + I) \log_2 n + K$ keys. Group multicast message size for each addition and leave operation is $2(\log_2 n - 1)(K + I)$.

The scheme also addresses basic requirements of group key management security which are listed in the following:

Backward Secrecy: In case of join at the time of $t=T$, group secret keys should be updated in such a way that added user should have no access to any key used in the group at the time $t<T$.

Forward Secrecy: In case of eviction from the group at the time of $t=T$, group secret keys known to the evicted user should be updated in such a way that evicted user should have no access to any key used in the group at the time $t>T$.

Collusion freedom: None of the evicted users or coalitions could generate the current or future group keys from the keys used in the past.

Chapter 2 gives the core idea of hierarchical binary tree model, secure group formation and key agreement protocols and performance evaluation and security analysis of the proposed protocols, respectively.

The proposed hybrid authenticated key establishment protocol in Chapter 3 improves the efficiency of the entity authentication and join process in EDKM. Hybrid cryptography is a combination of elliptic curve cryptography (ECC), secret key cryptography, modular squaring (as proposed in Modified Rabin's Signature scheme [55]) and one-way hash functions. Using hybrid combination provides a significant decrease in computational efforts and increased communication reliability with decreased message bandwidth.

Encryption link key establishment and mutual authentication issues in wireless sensor networks are addressed in Chapter 4. The proposed lightweight key establishment protocols meet the following expectations of sensor networks:

Dynamically relocating stations: Proposed schemes establish distinct keys for every session between any dynamically relocating data collecting station and the sensor in the same system cluster.

Mutual authentication: Sensor and data collecting station mutually authenticate each other before they initiate the secure data communication.

Lightweight operations on sensor side: Proposed protocols avoid using expensive modular exponential or elliptic curve point multiplications. Sensor needs to make only large or small modular multiplications, secret-key encryption/decryption and hashing operations.

Power efficient key establishment: While strong security advantage of public key cryptography is taken in proposed schemes, the minimum amount of battery power is consumed.

The public key encryption and digital signature algorithms can be introduced to Kerberos authentication protocol in several ways to prevent password-based attacks and to provide mutual authentication [39]. Using Elliptic Curve Cryptography (ECC) offers higher strength per key bit in comparison with other public key methods [2]. However without efficiently designed protocols elliptic curve operations take excessive amount of time and require bigger code size than mobile platform can usually offer. The major goals that the designed authentication protocol should meet are given in following:

- Preserve the main semantics of Kerberos.
- Provide mutual authentication between client and KDC.
- Provide non-repudiation of client to KDC.
- Minimize the number of operations to be performed on mobile client.

- Keep ability to use existing or developing public key management infrastructures.

Chapter 5 provides brief background on public-key enabled Kerberos protocol and proposes efficient ECC enabled Kerberos authentication protocol which meets expectations given above. Chapter 5 also gives the details of high speed and scalable implementation of the proposed protocols and specifications of used software and hardware.

CHAPTER 2

EDKM: EFFICIENT DISTRIBUTED KEY MANAGEMENT FOR MOBILE AD HOC NETWORKS

Özkan M. Erdem

School of Electrical Engineering and Computer Science

Oregon State University

Corvallis, OR 97331

Proceedings of IEEE International Symposium of Computers & Communications

ISCC'2004

Alexandria, Egypt, July 2004.

Abstract

Mobile ad hoc networks are dynamically reconfigured networks. Their major properties are mobility of devices, lack of central control authority, and existence of resource-constrained devices. Particularly, it is prudent to assume that there are no shared secret keys distributed by key distribution authority at initialization of the network. In this paper, we propose a new efficient hierarchical binary tree model (EDKM) to form ad hoc group under such assumptions. Our trust model employs a new key distribution scheme to bring an alien device to the group and to exchange a secret key at that moment. EDKM is distributed, self-organizing, and can be deployed incrementally in the network. Moreover, EDKM provides complete backward and forward security in case of modification in membership and does not increase processing or storage requirements in comparison with other HBT schemes.

The proposed group key management system is based on one-way hash function and secret key cryptography, and therefore, EDKM is practical, efficient and respectful to the constraints of mobile ad hoc networks.

2.1 Introduction

Ad hoc networks can be defined as the collection of devices where they do not have a fixed network topology [2]. Devices are often mobile and networking functions are performed by the devices in a self-organizing manner. Mobility causes ad hoc networks to be formed, partitioned into separate networks or merged with other networks. For this reason, traditional security models may not be fully applicable in ad-hoc networks.

Introducing security to mobile networks can be accomplished in two different methods. First method employs a single central authority which establishes the trust among devices, issues and manages keys and (if necessary) certificates [7]. However, major difference between ad-hoc networks and traditional networks is the lack of central control over the network structure and on-line availability of this trusted authority; therefore centralized security solution may not work for wireless ad hoc networks [14]. In second and widely accepted method security does not rely on any trusted authority or fixed central server; devices establish self-organized trust relation [2, 6, 14].

Hierarchical binary tree (HBT) is one of the previously studied techniques to manage the keys in distributed group structure [3, 4, 5, 8, 11]. Most of the relevant proposals involve a key distribution authority which maintains a tree of keys and manages the group structure. Each member of the tree holds its own key and corresponding ancestor keys. In addition, all group members share the group key provided by KDC. When the group structure changes, KDC initiates the key update procedures by sending encrypted messages to members. At this point, current proposals either suppose pre-established trust relation between KDC and members or employ public key operations to encrypt the messages.

In this paper, we propose a practically secure protocol to build efficient hierarchical binary tree trust model for mobile ad hoc networks which has no trusted third party to maintain the keys. Our proposal, which we call it as EDKM protocol, employs a secure distributed model based on secret-key encryption algorithm and one-way hash function. The EDKM protocol also includes group keying and group membership change operations (single join/leave, multiple join/leave). For a balanced tree associated to n users, each member holds $(2K + I) \log_2 n + K$ keys. Group multicast message size for each addition and leave operation is $2(\log_2 n - 1)(K + I)$.

We focus in the core idea of hierarchical binary tree model in Chapter 2.2. Secure group formation and key agreement protocols are presented in Chapter 2.3. Performance evaluation and security analysis of protocols are given in Chapter 2.4 and in Chapter 2.5, respectively.

2.1.1 Related Work

Although majority of research on security of ad hoc networks emphasize the secure routing protocols, there are some proposals on key generation and distribution issues. Zhou and Haas addressed security issues in ad hoc networks, and proposed security services based on public-key cryptography [14]. Another proposed protocol, called CLIQUES, makes improvement in the system capability to distribute session keys for dynamic groups [12]. CLIQUES uses Diffie-Hellman key establishment algorithm and group manager has to perform $O(n)$ exponentiations for each group addition and eviction. Authors assume each device in network has sufficient resources to run consecutive $O(n)$ public-key crypto operations.

The use of HBT (Hierarchical Binary Tree) in group key management has been proposed several times. In the proposed protocol by Wallner et al, each internal node

has a secret key and when the group membership changes each updated key is encrypted with each of its children's keys and multicast to children [13]. McGrew and Sherman made an improvement to the hierarchical binary tree approach in their OFT protocol and used one-way function tree to update the group keys [8]. In OFT, each node's key is blinded using one-way hash function and each node knows its ancestors' keys together with the blinded version of each ancestor's sibling key. In case of change in the group membership, manager encrypts the new value of node's blinded key with the blinded key of its sibling device and then multicast to appropriate members. Thus, messages for adding or evicting a member carry $\log_2 n$ keys. Perrig et al proposed another protocol, called ELK, which uses HBT and pseudo random number functions [10]. In their approach KDC refreshes the root key and updates the whole tree keys in every time interval using PRF function. Dondeti et al proposed a distributed group key management scheme for secure many-to-many communications, called DTKM [5]. In their approach, key associations are designed to delegate the task of key distribution evenly among the group devices. Updated keys are encrypted by ancestor keys and multicast to the other members by the members in key association. CTKM is an alternative HBT based protocol proposed by Caronni et al [3]. Similarly, this protocol uses one-way functions to form the keys in the group. However, in CTKM, all keying material is managed centrally by the Group Manager where all joining participants have to register and share the secret. Since Group Manager controls all the group keys, this brings a single point of attack and failure in the group.

2.2 Hierarchical Binary Tree Model

Hierarchical binary tree (HBT) consists of root node which locates at the top of the tree and leaf nodes locating at the lowest level *d*. Each node in the tree is either leaf

node or parent of two nodes (internal node) and each leaf node corresponds to one group member. Internal nodes hold their own keys and do not represent any member. We call the internal nodes in the path starting from member's parent up to the root as *ancestor nodes* and the set of ancestor nodes as *ancestor set.* Each internal node's key is derived from one of its children keys and each node contributes to generation of their ancestor node keys as long as node gets a new sibling or its sibling leaves the group. This also brings the contribution of sibling member to the group key which is the calculated final key of the root node.

The depth of the balanced binary tree is $d=\log_2 n$ where n is the number of group members. We assume a balanced HBT throughout this paper since it gives us more efficiency in number of computation and in size of messages required for member addition and eviction from the group. We don't employ any key distribution authority or group manager, so the members get together without shared secrets. Maintenance of the keys is totally performed by the group members in self-organizing manner. This feature separates our proposal from several other recent proposals based on HBT [2, 3, 7, 10].

Each node is assigned a unique binary identity *Id*, which is used for locating the node in the group. The length of the *Id* shows the depth of node. We illustrate the example HBT trust model among eight devices in **Figure 2-1**. The binary streams noted below the devices and above the internal nodes are uniquely assigned identity numbers.

Key Sharing Subgroup (KSS) is designed to provide key exchange between its members. The main purpose of using KSS is to provide peer-to-peer secure transfer of blinded key which is originally held by the sibling of ancestor nodes. When any member leaves the group, sibling uses blinded keys to encrypt new group keys before sending to group. Leaving member has no knowledge of blinded keys in use by its

sibling. Thus, forward secrecy is perfectly preserved. KSS contains member itself, sibling and one selected child/grandchild device of each ancestor's sibling.

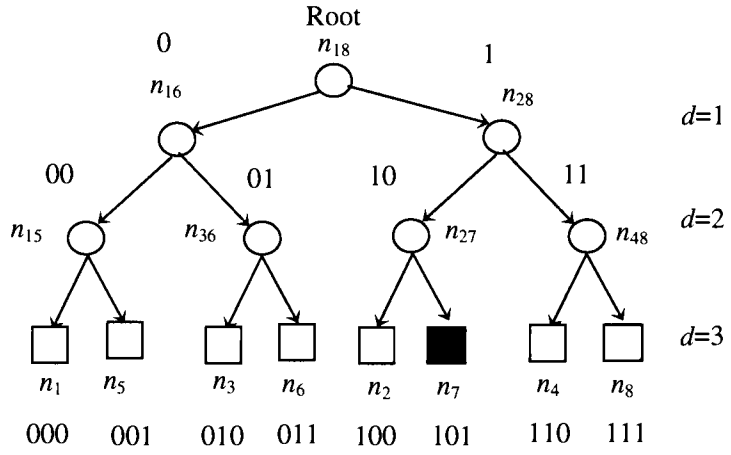

Figure 2-1 Hierarchical one-way function binary tree model

Blinded version of the original key k is simply the result of $BK(k, i) = h(k \oplus i)$ where h is a one-way hash function, $\oplus$ is a normal xor operator and i is the random index value. One way hash function hides the original value of k and i into $BK(k, i)$ in such a way that the knowledge of i and BK does not give the attacker any information about k.

We give the brief algorithm *Generate_KSS*() to find members of KSS for the member n in **Figure 2-2**. *Generate_KSS* algorithm returns the *Id* number of every member locating in its own KSS.

In EDKM, authentication is the first step for alien device to sign up the group. First, alien device chooses a sibling which already is a group member. Having agreed session encryption key, both devices begin to talk securely and alien device becomes the member of the group.

Regarding the example in **Figure 2-1**, n_7 (101) desires to join the group. When we apply the algorithm to generate KSS for n_7, we get the members n_2 (100), n_8 (111) and

n_5 (001). Member n_7 obtains blinded keys $BK(k_2, i_2) = h(k_2 \oplus i_2)$, $BK(k_{48}, i_{48}) = h(k_{48} \oplus i_8)$ and $BK(k_{16}, i_5) = h(k_{16} \oplus i_5)$ from these members respectively.

```
Generate_KSS (n):
i=1;
while (i <= depth_of_node (n))
{ Id_i = Id_n;
  exor i^th bit of Id_i with "1";
  if Id_i is leaf address then add Id_i to KSS (n);
  else if Id_i is internal address
    {while Id_i is not leaf address
       shift Id_i one bit left;
     add Id_i to KSS (n);}
  else
    {while Id_i is not leaf address
       shift Id_i one bit right;
     add Id_i to KSS (n);}
  increment (i);
}
```

Figure 2-2. Searching algorithm for members of Key Sharing Subgroup

Index values i_2, i_8 and i_5 should be sent separately in the messages. Since the message traffic is encrypted with previously agreed session key, neither other members nor intruders can access to blinded keys. Keys of ancestor nodes are generated by the formula of $k_i = h(h(\ldots h(k_{leaf})))$. Here k_{leaf} denotes the key which belongs to one member under this ancestor. In our proposal, after each change in group membership, sibling of added or evicted member renews its own key and updates its ancestor keys using the formula above. Since hashed values of the keys are computed in the upward direction, any member performing the calculation of k_i cannot retrieve the original value of k_{leaf}.

2.3 Group Formation and Operations

In this chapter, we define the protocols to authenticate the members to the group and describe the basic algorithms for join and leave operations.

A traditional way is to use public key cryptography for secret message encryption key establishment [2, 6, 7, 12]. We construct our group authentication protocols based on public key cryptography. Based on load-balancing requirement, the difference between the depth of the shallowest device and the depth of the deepest device should not be greater than two. Members can hold dynamically changing depth information of the shallowest device in the group as well as their own depth information. Thus, each member evaluates its own position availability to accept new sibling. The notation we used in this chapter is given in Table 2-1.

Table 2-1. Notation

Symbol	Meaning
AuthReq	Authentication request
AuthRes	Authentication response
KeyReq	Group key request
KeyRes	Response to key request
P_j	Public key of the user j
K_j	Secret key generation key of j
Id_j	Network identification of user j
GK	Group key
$Cert_j$	Public key certificate of user j
BK	Blinded key
$h(x \oplus y)$	One-way hash function output of x exor y
E [*SK*, *M*]	CBC mode encryption of the message M under the key *SK*
PuE [P_j, *K*]	Public key encryption of K with the user j's public key P_j
t_j	Time-stamp value of the message transmitted from user j
PdK	A key derived from group join password
MAC[*K*, *M*]	Message authentication code of M generated with key K
SK	Established secret encryption/decryption key between two users

n_s	n_a
	Generate random number r_a.
	Send $Cert_a$, r_a and MAC code generated with PdK.
	⟵
	AuthReq: M= ($Cert_a$, r_a) , MAC[PdK, M]
Receive.	
If $Cert_a$ or MAC is not verified, then quit.	
Assign identity Id_a.	
Generate random 128 bit K_s and encrypt K_s with public key P_a.	
Add time-stamp t_s.	
Generate MAC code of message using K_s.	
Send.	
⟶	
AuthRes: M = (Id_s, Id_a, $Cert_s$, PuE [P_a, K_s], t_s, r_a), MAC[K_s, M]	
	Receive.
	If r_a does not match to sent r_a, then quit.
	Decrypt K_s using private key S_a.
	If $Cert_s$ or MAC is not verified, then quit.
	Generate random 128 bit K_a and encrypt with public key P_s.
	Insert time-stamp t_a to the message along with t_s.
	Send.
	⟵
	KeyReq: M= (Id_a, Id_s, PuE [P_s, K_a], t_a, t_s), MAC[K_a, M]
Receive.	
If t_s does not match to sent t_s, then quit.	
Decrypt K_a using private key S_s.	
If MAC is not verified, then quit.	
Compute secret message encryption key; $SK = h\ (K_a \oplus K_s)$.	
Generate *KeyRes* message which consists of GK, BK, index i, ancestor keys (AK) and previously received time-stamp t_a.	
Encrypt the message, generate MAC code with SK.	
Send.	
⟶	
***KeyRes*:** M = (E[SK, (GK, BK, i, AK, t_a)]), MAC[SK, M]	
	Receive.
	If t_a does not match to sent t_a, then quit.
	Compute $SK = h\ (K_a \oplus K_s)$.
	If MAC is not verified, then quit.
	Decrypt the keys and store.

Figure 2-3. Group authentication protocol

Each device has its own public-private key pair (P_i, S_i) stored on the device. Authenticity and integrity of the identity information and public keys are preserved with X.509 standard certificates signed by Certification Authorities who are known to other group devices. These methods are out of scope of this paper and reader is referred to [9] for details. We assume each legitimate user has the knowledge of group password which is required for join and each user has public key certificate to prove identity prior to join.

Alien device sends a multicast join request to the group. Each member responds this message unless the difference between its own depth and the depth of the shallowest member is greater than one. This process helps us to build a load-balanced group after join. Alien device, n_a, picks one of the responded members, n_s, as its sibling. In proposed four-way protocol, devices simply exchange their public keys and establish secret message encryption key. Detailed group authentication protocol steps are presented in **Figure 2-3**.

2.3.1 Member Join Algorithm

Single Member Join: When a single device joins to group, authenticator member becomes sibling and moves one level down. Sibling also generates a new leaf and inserts it to the tree as a parent of both nodes.

In **Figure 2-4**, we envisage a mobile ad hoc network example formed by seven devices. When the device n_8 joins the group, n_4 becomes its sibling. Sibling n_4 generates the leaf n_{48} as a parent. Joined device n_8 finds out members of its KSS by executing *Generate_KSS*. Following message encryption secret key establishment with each KSS member respectively, n_8 requests blinded version of ancestor's

sibling's key from KSS members. Each KSS member generates distinct i index value, computes $h(k_{ancestor} \oplus i)$ and sends this hash value back to the joining device along i.

In our example, first KSS member n_6 generates i_6, computes $h(k_{16} \oplus i_6)$ and sends this value to n_8 together with i_6. Next KSS member n_2 generates i_2, computes $h(k_{27} \oplus i_2)$ and sends this value to n_8 together with i_2.These blinded keys are encryption keys to be used for eviction/leave process. Message traffic between n_2, n_6 and n_8 should be encrypted with previously established secret message encryption keys. Sibling (n_4) generates its own key k_4 and computes the key $k_{48}= h(k_4)$. Keys k_{28} and k_{18} are updated ($k_{28}'=h(k_{48})$, $k_{18}'=h(k_{28})$) by sibling, respectively. Updating the ancestor keys provides the backward secrecy. In order to inform other group members about modified ancestor keys, sibling sends modified keys encrypted with previously obtained blinded keys. Each member gets their modified ancestor key and blinded version of ancestor's sibling key with multicast messages. Multicast messages also include corresponding index values in plaintext format. For our example; n_4 multicasts E[$BK(k_{27})$, ($BK(k_{48})$, i_{48}, k_{28})], i_{27} to children of n_{27} and multicasts E[$BK(k_{16})$, ($BK(k_{28}')$, i_{28}, k_{18})], i_{16} to grandchildren of n_{16}.

The last duty of sibling is sending E[$SK_{n4,n8}$, ($BK(k_4)$, i_4, k_{48})] to new member n_4. In order to reduce the number of messaging, *KeyRes* message consists of this encrypted transmission. Joining member calculates upper ancestor keys and group key by consecutive hashing operations.

Multiple Member Join: We suppose that group has n members and multiple join operation is the mass join of i devices such that i is between $n/2$ and n. During mass join, sibling of each joined member individually updates its ancestor keys and multicasts the blinded keys and ancestor keys to the group. Final values of ancestor keys are computed exor values of received ancestor keys. New root key is assigned as

a group key. Since each sibling sends (d-1) different blinded keys and ancestor keys to the group, the number of required computations increases linearly with the number of joining device.

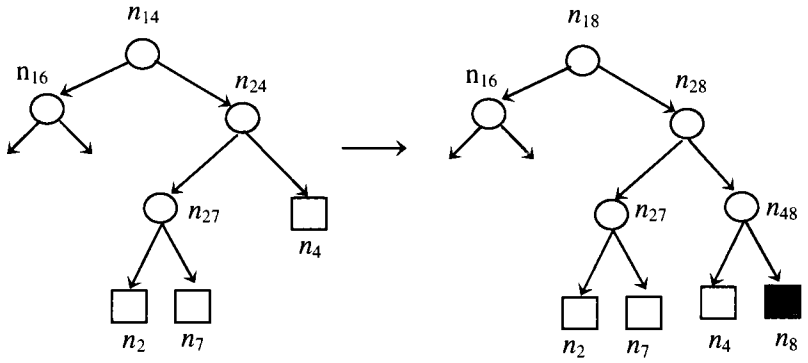

Figure 2-4. Single member (n_8) joins the group

2.3.2 Member Leave Algorithm

Single Member Leave: Modification of the keys in member eviction is very similar process to the modification in join. Key modifications are executed by the sibling of leaving or evicted device. Sibling member removes the parent leaf and locates to parent's place. To preserve the forward secrecy, sibling renews the ancestor keys and so the group key.

As illustrated in **Figure 2-5**, n_2 leaves the group and sibling n_7 replaces its parent. Member n_7 generates k_7 and updates ancestor keys; $k_{78} = h(k_7)$ and $k_{18}' = h(k_{78})$. Since other group members do not have the renewed keys, n_7 multicasts E[$BK(k_{48})$, ($BK(k_7)$, i_7, k_{78})], i_{48} to the children of n_{48}.

Member n_7 also multicasts E[$BK(k_{16})$, ($BK(k_{78})$, i_{78}, k_{18})], i_{16} to the grandchildren of n_{16}. Each device computes the renewed upper ancestor keys by applying consecutive hash operations to received ancestor key.

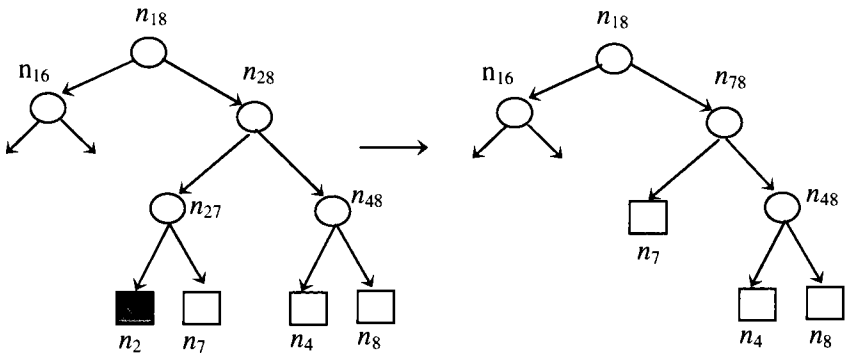

Figure 2-5. Single member (n_2) leaves the group

In our example, children of n_{48} compute $k_{18}' = h(k_{78})$. Blinded key of n_7, $BK(k_7)$, is used to inform n_7 by the children of n_{48} about future key modifications in a secure way. Moreover, members locating under n_{16} use $BK(k_{78})$ to inform members locating under n_{78} for future key modifications securely.

Multiple Member Leave: In case of mass leave, stayed members modify all ancestor keys in a similar manner. Suppose that n_2 and n_8 leave the group. Leaf n_{27} and its sibling n_{48} are removed from the group. Leaves n_{28} and n_{18} are renamed to n_{74} and n_{14}, respectively. Each sibling individually updates the ancestor keys k_{74} and k_{14}. Final values of k_{74} and k_{14} are computed with very same process mentioned in multiple member join; $k_{74} = k_{74}' \oplus k_{74}''$ and $k_{14} = k_{14}' \oplus k_{14}''$ where k_{74}' and k_{14}' represent the keys updated by n_7. Moreover, the keys k_{74}'' and k_{14}'' represent the updated ancestor keys by n_4. Multicast messages for blinded keys to other group members are transmitted as specified in single leave algorithm.

2.4 Performance Comparisons

In this chapter we analyze the EDKM protocol in terms of computational time and message size requirements. We have selected four different group key management schemes for comparisons. The notation we used throughout the chapter is shown in **Table 2-2**.

In **Table 2-3**, we present the total size of unicast messages, size of multicast messages, required computation from KDC and from group members for single and multiple join case. In multiple join, we consider the worst case scenario which is adding n new users to the group with n users. This scenario is accomplished with adding new sibling to every current member. In equations, d shows the depth of the tree before joining process.

Table 2-2. Notation

Symbol	Meaning
PE	Public key encryption
PD	Public key decryption
E	Secret key encryption
D	Secret key decryption
d	Depth of the tree
I	Size of index in bits
K	Size of key in bits
H	Hash function
G	Key generation
X	Xor operation
KS	Key setup between member and group manager
n	Number of joining or leaving devices

In **Table 2-4** we summarize the case of single and multiple eviction from the group. We focus on worst case eviction scenario. Group has originally $2n$ members and half of the members leave the group. In other words, sibling of each member leaves the group and so the depth of the tree decreases by 1.

Table 2-3. Comparison of single and multiple join equations

		Message Size	
		Unicast	Multicast
EDKM	S. Join	$d(K+I)+K$	$2(d-1)(K+I)$
	M. Join	$n: d(K+I)+K$	$n: 2(d-1)(K+I)$
OFT	S. Join	$(d+3)K+I$	$(d+1)K$
	M. Join	$n: (d+1)K+I+2K$	$(2n-2)K$
ELK	S. Join	$(d+1)K+I$	-
	M. Join	$n: (d+1)K+I$	-
DTKM	S. Join	$2dK$	$(d-1)K$
CTKM	S. Join	dK	dI
		Computation	
		KDC	Members
EDKM	S. Join	-	$G+d(2E+2H+X+D)+H-E$
	M. Join	-	$nG+d(2E+D+2H+X)+(2n-d-1)X+H-E$
OFT	S. Join	$G+(d+1)H+d(X+3E)$	$(d+3)D+2d(H+X)$
	M. Join	$nG+(4n-2)(H+X)+(dn+5n-1)E$	$(d+3)D+2d(H+X)$
ELK	S. Join	$G+(4n-2)E+(d+3)E$	$(d+1)D+2dE+2E$
	M. Join	$(8n-2)E+nG+n(d+3)E$	$(d+1)D+(2d+2)E$
DTKM	S. Join	-	$2d(PE+PD)+(d-1)E+d(X+H)$
CTKM	S. Join	(GM): $KS+E+dH$	$KS+D$

DTKM [5] and CTKM [3] are examples of schemes which do not require to use trusted third party, whereas OFT [8] and ELK [10] require to have real time key distribution authority. As seen in **Table 2-3**, EDKM requires approximately twice more unicast messaging than OFT and ELK require for joining operations. That is because each member sends d authentication unicast messages during join. However, EDKM performs better as for the number of required computations than the total computations in OFT and ELK. **Table 2-4** shows us that EDKM loses out to OFT and ELK in comparison of multicast message size for leaving operations. As an obvious reason, sibling of every leaving member sends d-1 multicast messages to update the keys in EDKM. Note that in the ELK protocol, we have chosen $n_1+n_2 = K$ to provide

the same level security with other schemes. When it comes to the number of required computations for leaving operations, EDKM again achieves better results than the other protocols.

Table 2-4. Comparison of single and multiple leave equations

		Message Size	
		Unicast	Multicast
EDKM	S. Leave	-	$2(d - 1)(K + I)$
	M. Leave	-	$n : 2(d - 1)(K + I)$
OFT	S. Leave	-	$I + (d + 1)K$
	M. Leave	-	$nI + (3n - 2)K$
ELK	S. Leave	-	$I + d(n_1 + n_2)$
	M. Leave	-	$nI + (n - 1)(n_1 + n_2)$
DTKM	S. Leave	$(d - 1)K$	$(d - 1)K$
CTKM	S. Leave	$(2d - 1)K$	-
		Computation	
		KDC	Members
EDKM	S. Leave	-	$G + (d - 1)(E + X + H) + dH$
	M. Leave	-	$nG + d(2H + E + X) - (E + H) + (2n - d - 2)X$
OFT	S. Leave	$D(E + H + X)$	$D + d(H + X)$
	M. Leave	$(2n - 2)H + (n - 1)X + (3n - 2)E$	$(d + 1)D + d(H + X)$
ELK	S. Leave	$8dE$	$dD + 5dE$
	M. Leave	$(7n - 3)E$	$dD + 5dE$
DTKM	S. Leave	-	$(d - 1)(PD + PE + E + X + H)$
CTKM	S. Leave	(GM): $G + (2d - 1)E$	-

2.5 Security Analysis

The security of the EDKM group key management protocol relies on random selection of private key at user side and the cryptographic security of used hash function. Although there is no successful attack on full SHA1 and MD5 algorithms, one-way hash functions are not theoretically secure [1]. However, since the best method known to break full hash algorithm is brute-force attack we accept one-way hash functions are computationally secure [9]. EDKM provides the following security features:

(1) *Backward secrecy*: Sibling of joining member generates new private key and modifies the keys of ancestors with consecutive hashing operations applied to new private key. New keys of the branch are transmitted to the other members after encryption by blinded keys of ancestor's siblings, respectively. Since group key and all keys on the branch are modified, joining device cannot access to any key used in the past.

Each blinded key is specifically generated for only one member of the group. Index values should not be repeated for blinding the same key. The security of blinded key relies on the strength of mixing function (exor) and one-way hash function.

(2) *Forward secrecy*: Evicted member does not have the knowledge of blinded keys that its sibling holds. Since all ancestor keys are renewed by the sibling after eviction process, evicted user cannot access to future group communication keys.

(3) *Group confidentiality*: Nonauthorized users do not have access to any key used in group communications unless they prove their credentials and authenticate to group.

(4) *Free of collusions*: Previous keys are not used in derivation of new keys. So, any cooperation of evicted users cannot recover the future keys from the keys used in the past.

2.6 Conclusions

In this paper, we presented and analyzed a framework for group key management in mobile ad hoc networks. This framework consists of hierarchical binary tree based group key distribution.

Our efficient hierarchical binary tree protocol, called EDKM, uses one-way hash function and simple xor operations. EDKM achieves comparable overall performance with the proposals which use secret-key cryptography and employ key distribution authority.

Consequently, EDKM offers practical distributed key management for resource constrained systems which has lack of key distribution authority.

2.7 References

[1] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Cryptographic hash functions: A survey. Technical Report 95-09, University of Wollongong, July 1995.

[2] S. Capkun, L. Buttyan, and J. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52-64, January-March, 2003.

[3] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner. Efficient security for large and dynamic multicast groups. *Proceedings of the Seventh Workshop on Enabling Technologies (WET ICE'98)*, IEEE Computer Society Press, 1998.

[4] R. Di Pietro, L. V. Mancini, and S. Jajodia. Efficient and secure keys management for wireless mobile communications. *Proceedings of POMC'02, ACM*, Toulouse, France, October 2002.

[5] L. R. Dondeti, S. Mukherjee, and A. Samal. A distributed group key management scheme for secure many-to-many communication. *Technical Report PINTL-TR-207-99*, Department of Computer Science, University of Maryland, 1999.

[6] S. Gokhale and P. Dasgupta. Distributed authentication for peer-to-peer networks. *International Symposium on Applications and the Internet (SAINT) 2003*, pages 347-353, Orlando, FL.

[7] A. Khalili, J. Katz, and W. Arbaugh. Toward secure key distribution in truly ad-hoc networks. *International Symposium on Applications and the Internet (SAINT) 2003*, pages 342-346, Orlando, FL, 2003.

[8] D. A. McGrew and A.T. Sherman. Key establishment in large dynamic groups using one-way function trees. *Technical Report No.0755*, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[9] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of applied cryptography. *CRC Press series on discrete mathematics and its applications*. CRC Press, 1997.

[10] A. Perrig, D. Song, and J. D. Tygar. ELK, a new protocol for efficient large-group key distribution. *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2001.

[11] S. Rafaeli, L. Mathy, and D. Hutchison. EHBT: An efficient protocol for group key management. *Proceedings of the Third International COST264 Workshop (NGC 2001)*, Springer-Verlag, LNCS 2233, pages 159-171, London, UK, November 2001.

[12] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A new approach to group key agreement. *Technical Report RZ 2984*, IBM Research, December 1997.

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

[14] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6), November-December 1999.

CHAPTER 3

EFFICIENT SELF-ORGANIZED KEY MANAGEMENT FOR MOBILE AD HOC NETWORKS

Özkan M. Erdem

School of Electrical Engineering and Computer Science

Oregon State University

Corvallis, OR 97331

Proceedings of IEEE Global Telecommunications Conference

Globecom' 2004 Security and Network Management Symposium

Dallas, TX, USA, December 2004.

Abstract

In mobile ad hoc networks, authentication of users and management of group keys require more specialized solutions than traditional security protocols. Mobility of the users and unavailability of trusted central servers are major known issues for mobile networks. In this paper, an efficient and practical solution has been proposed for group key management problem. Members authenticate each other with fast and efficient hybrid key establishment scheme which combines elliptic curve cryptography, modular squaring operations and secret key encryption algorithm. The proposed group key management model is based on hierarchical binary trees. Users exchange the group secret keys and form an ad hoc group in distributed and self-organizing manner with no central control. This paper also discusses the total backward and forward security in case of modification in membership. Finally, the proposed group key management system is scalable and respectful to the constraints of ad hoc networks. It also has comparable efficiency to the other HBT based key management protocols which employ real time key distribution authority.

3.1 Introduction

Ad hoc network devices are usually mobile and the devices perform networking functions in a self-organizing manner. Mobility changes the topology of the network frequently; nodes can relocate often, leave one ad-hoc group and join to another group rapidly. Due to mobility of devices, traditional security models designed for fixed-network topologies may not be fully applicable in ad-hoc networks. Securing mobile ad hoc networks has brought new challenges and opportunities in terms of trust model, key management, and the standard security services such as confidentiality, authentication and integrity.

The problems of securing the communication between mobile devices and management of group confidentiality keys are studied in this paper. Recent studies introduce three different general key management schemes to mobile networks. The trusted central authority scheme employs single or multiple servers to establish the trust among devices and to manage the keys [3, 10]. Kerberos is a well-known example of this method [7]. However, because of on-line availability issue of this trusted server and lack of central control over the mobile network structure, centralized security solution may not work for ad hoc networks [15]. In second type of scheme, key information is distributed among devices prior to forming the group. This scheme is especially suitable for computationally weak devices. However, pre-distribution of keys can be performed only if devices have prior trust relationship. Besides, memory and bandwidth efficiency in this scheme significantly decreases when the number of devices increases. In a third and widely accepted method, security does not rely on any trusted authority or fixed central server; devices establish self-organized trust relations using asymmetric cryptography [4, 15]. The use of public key techniques along with the key certificates can help to overcome the difficulty of peer-

to-peer key establishment. Also, compromise of any device does not pose a security risk to traffic between other devices. There are some recent proposals based on modular exponentiation techniques for the management of encryption keys. However, portable devices usually have constraints in computational capability, bandwidth and battery power to afford computationally expensive operations. Using modular exponential techniques may not be possible for the network formed by resource constrained portable devices.

In this paper, hybrid authenticated key establishment protocols and practically secure and reliable group key management scheme are proposed. Hybrid cryptography is a combination of elliptic curve cryptography (ECC), secret key cryptography, modular squaring (as proposed in Modified Rabin's Signature scheme [14]) and one-way hash functions. Using hybrid combination provides a significant decrease in computational efforts required to encrypt/decrypt messages and increased communication reliability with decreased message bandwidth.

The proposed group key management scheme builds hierarchical binary tree trust model to form the group, which has no trusted authority to maintain the keys. The proposed scheme also includes group re-keying operations in case of modifications in membership (member join and member leave). Basic confidentiality services provided by the scheme are listed in the following:

Backward and Forward Secrecy: In case of join at the time of $t=T$, group secret keys should be updated in such a way that added user should have no access to any key used in the group at the time $t<T$. In case of eviction from the group at the time of $t=T$, group secret keys known to the evicted user should be updated in such a way that evicted user should have no access to any key used in the group at the time $t>T$.

Collusion freedom: None of the evicted users or coalitions could generate the current or future group keys from the keys used in the past.

3.1.1 Related Work

Usage of ECC has been proposed several times in wireless key agreement protocols. ECMQV protocol with implicit certificates [5], the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) protocol [12] and Aydos-Y-K key agreement protocol [1] are major ECC based key agreement schemes. However, all of the schemes mentioned above require at least three expensive point multiplication operations on each side if entity and key authentication are provided.

Hierarchical binary tree (HBT) is one of the previously studied techniques to manage the keys in distributed group structure [4, 6, 8, 11]. Most of the relevant proposals involve a key distribution center (KDC), which maintains a tree of keys and manages the group structure. McGrew and Sherman used one-way function tree to update the group keys in OFT protocol [8]. In OFT, each node's key is blinded using one-way hash function. In case of change in the group membership, manager encrypts the new value of node's blinded key with the blinded key of its sibling node and then multicast to appropriate members. Perrig et al proposed another protocol, called ELK, which uses HBT and pseudo random number functions [10]. In their approach KDC refreshes the root key and updates the whole tree keys in every time interval using PRF function.

Dondeti et al proposed a distributed group key management scheme for secure many-to-many communications, called DTKM [6]. DTKM protocol exchanges the keys between members of a key association group using public key cryptography. Updated keys are encrypted by ancestor keys and multicast inside of the key association group.

CTKM is an alternative HBT based protocol proposed by Caronni et al [4]. However, in CTKM, the Group Manager manages all keying material centrally where all joining participants have to register and share the secret. GM (Group Manager) controls all the group keys, and this makes GM a single point of attack in the group.

3.2 Hierarchical Binary Tree Model

Hierarchical binary tree (HBT) consists of a root node, internal nodes and leaf nodes at the lowest level d. Each leaf node corresponds to one group member and carries a personal secret key. Internal nodes do not represent any user. Internal nodes in the path starting from leaf (member)'s parent up to the root are called as *ancestor nodes* and the set of ancestor nodes is called as *ancestor set.*

In case of modification in membership, a selected member becomes a sibling of joining or leaving member. Sibling renews its own personal secret key and applies consecutive hashing operations to renewed key in upward direction to refresh ancestor keys. Calculated root key becomes a new group encryption key.

The depth of the balanced binary tree is $d = \log_2 n$, where n is the number of group members. A balanced HBT is assumed throughout this paper since it gives more efficiency in terms of required computations and required message bandwidth for member addition and eviction from the group. The proposed scheme does not employ any key distribution authority or group manager; members get together without shared secrets. Maintenance of the keys is totally performed by the group members in self-organizing manner. This feature separates the proposed scheme from several other schemes based on HBT.

3.3 Authenticated Hybrid Key Establishment Protocol

Authenticated hybrid key establishment protocol uses hybrid cryptography which comprises modular squaring, elliptic curve point multiplication, one-way hash function and secret key cryptographic algorithm. Using elliptic curve cryptography has advantages of smaller key lengths, faster processing speed and less communication overhead which exactly fit to requirements of mobile network formed by low-powered devices. Every user chooses a public-private key pair and has the public key certified by any certification authority before joining the mobile ad hoc group. Verification of implicitly certified key consumes less bandwidth and computational power. Therefore, public keys are implicitly certified by certification authorities in one-time off-line process.

Users select and share an elliptic curve E defined over $GF(p)$, a base point P and large order s. Certification authority CA chooses m-bit random prime numbers f and g, which provide the condition of $f \equiv 3 \pmod 8$ and $g \equiv 7 \pmod 8$. CA's public key modulus for the Modified Rabin's Signature algorithm is $n_{CA} = fg$ and private key is $d_{CA} = (n_{CA} - f - g + 5)/8$.

Each user U chooses a random private key $d_U \in [2, s - 2]$ and computes the public key $Q_U = d_U \times P$. User U provides Id_U and Q_U to CA. This out-of-band communication should be performed securely. Following the verification of user identity, CA generates a time-stamp value t_U and computes the hash values of Id_U and t_U, respectively. Next, CA computes the xor of $(Y_U = h(Id_U) \| h(t_U))$ and $(Q_U = Q_U.x \| Q_U.y)$, where h shows one-way hash function. Previously identified redundancy function R is applied to the result of $(Q_U \oplus Y_U)$. This step is crucial in Rabin's scheme to prevent existential forgery attacks. Thereafter, CA encrypts the output of $R(Q_U \oplus$

Y_U) with its private key d_{CA} and gets the public key constructor P_U of user U; ($P_U = (R (Q_U \oplus Y_U))^{dCA} \bmod n_{CA}$).

CA finalizes the certification process by sending (Id_U, P_U, t_U) back to the user U. Having certification response message, U computes $Y_U = h(Id_U) \parallel h(t_U)$ and decrypts P_U by squaring P_U modulo n_{CA} ($S_U = P_U^{\,2} \bmod n_{CA}$). After applying reverse redundancy function R^{-1} to S_U, U accepts the received implicitly certified public key only if the result of ($R^{-1}(S_U) \oplus Y_U$) is equal to its own public key Q_U.

3.3.1 Authentication to Mobile Ad Hoc Group

Every joining user has to prove its identity and the knowledge of group password to a selected group member, called *Authenticator*. Process begins with multicast join request message sent to the group. Based on load-balancing requirement, each member in HBT responds this message unless the difference between its own depth and the depth of the shallowest member is greater than one. Members can hold dynamically changing depth information of the shallowest node in the group as well as their own depth information. Joining user picks one of the responded members as Authenticator and initiates the authenticated key establishment algorithm.

Definition 1: Blinded version of the key k is the result of $BK(k, i) = h(k \oplus i)$ where h is a one-way hash function, $\oplus$ is a normal xor operator and i is the random index value. One way hash function hides the original value of k and i into $BK(k, i)$ in such a way that the attacker cannot recover k with the knowledge of i and BK.

Each group member knows a personal secret key, blinded sibling key, original ancestor keys and blinded ancestors' sibling keys.

Definition 2: A *Key Sharing Subgroup* (*KSS*) of member U is a subset of group members in which every member establishes secret session keys and shares blinded keys with the user U.

Authenticator sends the list of its own *KSS* members along with their public keys to joining user during authentication.

The main purpose of using *KSS* is peer-to-peer secure transfer of blinded keys. When any member leaves the group, sibling encrypts new group keys using previously obtained blinded keys before sending out. Leaving member has no knowledge of blinded keys that sibling holds. Thus, forward secrecy is perfectly preserved.

Mutual authentication and hybrid key establishment algorithm between new user U and authenticator A is illustrated in **Figure 3-1**. The protocol steps are executed as in follows:

i. User U generates *Authentication Request* (*AuthReq*) message which contains $M = (Id_U \parallel P_U \parallel t_U)$ and MAC value of M which is generated by group password *Pwd*.

ii. A checks the validity of time-stamp t_U; if t_U is not valid, then A aborts the protocol. A also verifies the MAC value of M by using group password *Pwd* as a MAC key. If MAC value is not valid, then A aborts the protocol. Next, A decrypts P_U and extract the hidden public key Q_U of U. Decryption is a simple operation which is squaring P_U modulo n_{CA} and then applying inverse redundancy function to the output S_U. A computes $(Q_U = S_U^{'} \oplus Y_U)$ to extract the public key Q_U. The protocol assumes that both users make equal contributions to session encryption keys. A generates random k-bit number r_A as its own contribution. Thereafter, A sends its own identity Id_A, public key constructor P_A, time-stamp t_A and generated r_A back to U in *Authentication Response* (*AuthRes*) message.

iii. U checks the validity of t_A which is received in *AuthRes* and aborts the protocol in case of failure in verification. Then, U recovers the hidden public key of A; applies inverse redundancy function to decrypted P_A and then computes $Q_A = S_A' \oplus Y_A$ where $Y_A = h(Id_A) \parallel h(t_A)$.

iv. Both users derive the shared key as in follows: User U performs an elliptic curve point multiplication $G = d_U \times Q_A = d_U \times d_A \times P$, where user A performs a similar type of multiplication by using its own private key d_A and Q_U; $G = d_A \times Q_U = d_A \times d_U \times P$. Shared key Gx is the most significant 128 bits in x coordinate of point G.

v. User U generates a random k-bit number r_U as its own contribution to the session encryption key. This contribution value is concatenated to r_A and encrypted using secret key encryption algorithm and the shared key Gx. Encrypted message is sent back to A in *Key Confirmation Request* (*KeyConfReq*) message.

vi. A decrypts the received message using secret key decryption algorithm and the key Gx. Next, A verifies the value of r_A and recovers r_U in the decrypted message. If verification of r_A fails, then A aborts the protocol.

vii. Hybrid key establishment protocol proposes to generate two different keys, each in 128 bits length. Session link key (*SKey*) is used to encrypt the session communication and *MacKey* is used to provide message authentication code along with MAC algorithm. Both parties derive *SKey* by concatenating the most significant 64 bits of each shared value, and hashing the output as seen in the formula; $h(MSB_{64}(r_U) \parallel MSB_{64}(Gx) \parallel MSB_{64}(r_A))$. The most significant 128 bits of the hash result form the *SKey*. *MacKey* is generated by slightly different technique; both users concatenate the least significant 64 bits of each shared number and hash the result as given in the following formula; $h(LSB_{64}(r_U) \parallel LSB_{64}(Gx) \parallel LSB_{64}(r_A))$. Here, the most significant 128 bits of hash output form the *MacKey*.

viii. Authenticator A moves one level down in tree and becomes the sibling of new member U. New node is generated and inserted to the tree as their parent. For the sake of backward secrecy, all ancestor keys and group key must be changed. A renews its k-bit personal secret key k_A and computes its ancestor keys with consecutive hashing operations applied to k_A in upward direction. For instance, parent node's key is computed as $k_{AU} = h(k_A)$. Grandparent's key is calculated as $h(h(k_A))$, and so on. Computed root key is used as a new group key GK. A also generates blinded k_A with randomly chosen is-bit length index i_A.

ix. A encrypts $(KSS(A) \parallel BK(k_A, i_A) \parallel i_A \parallel k_{AU})$ by $SKey$ and computes the MAC value by using $MacKey$. Then, A sends encrypted and authenticated message back to the user U. $KSS(A)$ contains $(Id \parallel P \parallel t)$ information for each former sibling of A. Thus, each new user gets d-1 different $(Id \parallel P \parallel t)$ entries for d-1 members of KSS, where d denotes the depth of tree after new join.

x. After successful verification of MAC value, U decrypts the encrypted message and obtains $KSS(A)$, $BK(k_A , i_A)$, i_A and parent key k_{AU}. Then, it generates a random k-bit personal secret key k_U and computes the ancestor keys and group key GK by applying consecutive hashing operations to k_{AU}. U also adds the entry of $(Id_A \parallel P_A \parallel t_A)$ to $KSS(A)$ and stores the new list as $KSS(U)$. Authenticator A adds the entry of $(Id_U \parallel P_U \parallel t_U)$ to $KSS(A)$ and stores.

User (U)	***Authenticator (A)***
Size k, index size is, modulus n_{CA} and password Pwd are shared.	
Send ***AuthReq***: $\{M=(Id_U \| P_U \| t_U)$, MAC$[Pwd, M]\}$ $\longrightarrow$	Receive
	If t_U is not valid, then abort.
	If MAC is not valid, then abort.
	Compute $Y_U = h(Id_U) \| h(t_U)$
	$S_U = P_U^{\,2} \bmod n_{CA}$. $S_U' = R^{-1}(S_U)$
	Compute $Q_U = S_U' \oplus Y_U$
	Generate random k-bit r_A.
	AuthRes: $\{Id_A, P_A, t_A, r_A\}$
Receive $\longleftarrow$	Send ***AuthRes***
If t_A is not valid, then abort.	Compute $G = d_A \times Q_U$
Compute $Y_A = h(Id_A) \| h(t_A)$	$Gx = MSB_{128}(G.x)$
$S_A = P_A^{\,2} \bmod n_{CA}$. $S_A' = R^{-1}(S_A)$	
Compute $Q_A = S_A' \oplus Y_A$	
Compute $G = d_U \times Q_A$	
$Gx = MSB_{128}(G.x)$	
Generate random k-bit r_U.	
Send ***KeyConfReq***: $c=E[Gx, r_U \| r_A]$ $\longrightarrow$	Receive
	$D[Gx, c] = r_U \| r_A$
	If r_A is not present, then abort.
	$SKey = MSB_{128}(h(MSB_{64} r_U \| MSB_{64} Gx \| MSB_{64} r_A))$
	$MacKey = MSB_{128}(h(LSB_{64} r_U \| LSB_{64} Gx \| LSB_{64} r_A))$
	Generate a new k-bit k_A and random is-bit index i_A
	Compute Ancestor Keys and GK
	$ms = E[SKey, KSS(A) \| BK(k_A, i_A) \| i_A \| k_{AU}]$
	KeyConfRes: $\{ms, mv=MAC[MacKey, ms]\}$
Receive $\longleftarrow$	Send ***KeyConfRes***
Compute $mv' = MAC[MacKey, ms]$	
If mv mv, then abort.	
$D[SKey, ms] = KSS(A) \| BK(k_A, i_A) \| i_A \| k_{AU}$	
Generate a new k-bit k_U	
$KSS(U) = KSS(A) \| (Id_A \| P_A \| t_A)$	$KSS(A) = KSS(A) \| (Id_U \| P_U \| t_U)$

Figure 3-1. Authentication to the mobile ad hoc group

3.3.2 Key Establishment with KSS Members

After the completed sign up process, new user U recovers the public key Q_V of every user V locating in its *KSS*. The method to recover Q_V is the same method used for recovering the public key Q_A of authenticator A. Then, user U initiates a simplified authenticated key establishment protocol with each user V in $KSS(U)$. This simplified protocol is shorter than the protocol given in previous chapter.

As illustrated in **Figure 3-2**, protocol assumes that user U has already recovered the Q_V and starts by sending its public key constructor. Thus, the real-time communication consumes less message bandwidth and requires smaller number of computations. This is especially beneficial when the protocol has to repeat d-1 times to establish secret keys with d-1 different users. After completion of each secret key establishment protocol, user U securely gets blinded key of ancestor's sibling and corresponding blinder index from each *KSS* member.

The last phase in user join is informing other group members about modified keys. In order to do this, authenticator multicasts the modified keys in encrypted format. Encryptions are performed using blinded keys, which were obtained when authenticator joined to group. Each member gets modified ancestor keys and blinded ancestor's sibling key with multicast messages. Multicast messages also include corresponding index values in plaintext format.

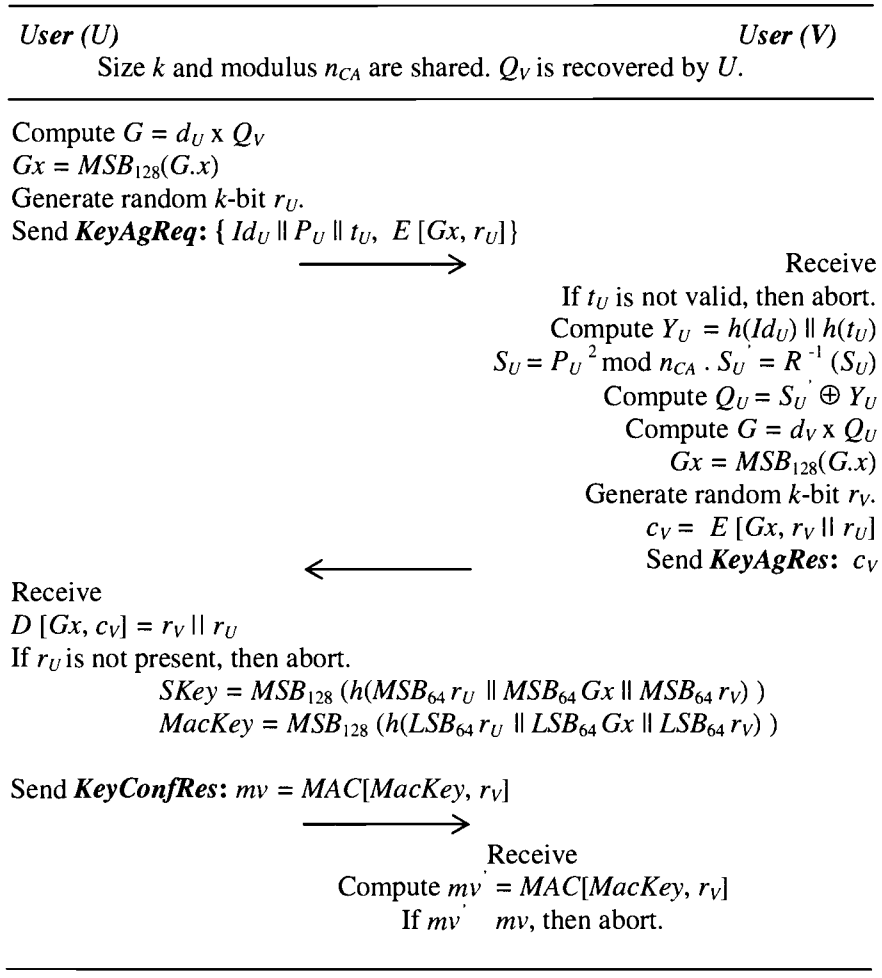

Figure 3-2. Authenticated key establishment with *KSS* members

As an example, n_5 joins the group in **Figure 3-3**. First, authenticator n_2 renews its own key k_2 and refreshes the ancestor keys: $k_{25}=h(k_2)$, $k_{24}'=h(k_{25})$ and $k_{14}'=h(k_{24}')$. Then, it sends new parent key k_{25}, blinded k_2 and index i_2 to n_5 encrypted by established secret key. Later, n_5 computes $k_{24}'=h(k_{25})$ and $k_{14}'=h(k_{24}')$. It also establishes distinct secret keys with n_1 and n_4, which are the members of $KSS(n_5)$. Finally, n_5 securely gets blinded k_{13} from n_1 and blinded k_4 from n_4.

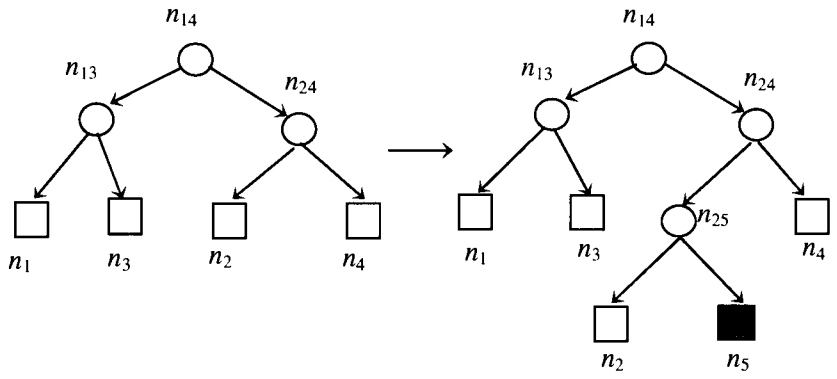

Figure 3-3. User n_5 joins the group

Since $KSS(n_2)$ contains n_1 and n_4, blinded keys obtained from these users should be used to encrypt multicast messages. Thus, n_2 sends $E[BK(k_4, i_4), k_{24}' \| BK(k_{25}, i_2) \| i_2]$ with index i_4 to n_4 and multicasts $E[BK(k_{13}, i_1), k_{14}' \| BK(k_{24}', i_2) \| i_2]$ with index i_1 to children of n_{13}, where $E[k, M]$ refers to secret key encryption of M using the key k. Other members simply use index came along with message to find the corresponding blinded key and to decrypt the message. Since each message carries only one modified ancestor key, rest of ancestor keys in upward direction and root (group) key are computed by applying consecutive hashing operations to the received ancestor key. Back in the example; n_4 needs to apply hash function to k_{24}' to find the new value of group key, k_{14}'.

3.3.3 Member Leave Algorithm

In case of member leave, sibling of the leaving user removes the parent node and locates to parent's place. However, since the leaving member knows the group key and its ancestor set keys, it can easily read the future group messages. Forward

secrecy is preserved by changing all the keys that leaving user knows. As in join algorithm, sibling of leaving user manages all necessary key modifications.

When n_2 leaves the group, n_5 replaces its parent and renews its own key k_5. It also refreshes ancestor keys using hash functions; $k_{54} = h(k_5')$ and $k_{14}' = h(k_{54})$. Later, it provides modified keys to other members in a secure way. Blinded keys, which are obtained during join of sibling, are used to encrypt the modified keys before transmission. First, n_5 generates the blinded versions of its new key k_5' and parent key k_{54} using randomly generated index i_5. Then, it sends $E[BK(k_4, i_4), k_{54} \| BK(k_5', i_5) \| i_5]$ with index i_4 to n_4, and the message $E[BK(k_{13}, i_1), k_{14}' \| BK(k_{54}, i_5) \| i_5]$ with index i_1 to the children of n_{13}. As in join algorithm, each member decrypts messages using index values and computes the refreshed ancestor keys in upward direction by applying consecutive hash operations to received ancestor key. In the example, n_4 computes $k_{14}' = h(k_{54})$. As a last step, every group member who has the entry of $(Id_{n2} \| P_{n2} \| t_{n2})$ in its *KSS*, replaces it with $(Id_{n5} \| P_{n5} \| t_{n5})$.

3.4 Security Evaluation

Authentication of user is accomplished by using implicit certificate which binds the public key of user to the unique identity. Note that, public keys can be calculated only by using authentic public modulus n_{CA}. Therefore, man-in-the-middle type attacks are mostly prevented. Moreover, implicit key confirmation is provided in protocol by added *KeyConfReq* and *KeyConfRes* messages. The final established keys are derived by equal contribution from each user. Thus, no single user has the entire control on derivation of secret keys.

Since the security of overall scheme also relies on secret-key encryption, encryption algorithm must be properly implemented. In case of using ECB encryption mode for

KeyConfReq, attacker is able to perform modification attack by concatenating fabricated value of $E[Gx, r_U]$ to the true value of $E[Gx, r_A]$ and resending back to A as if it is originally coming from U. Cipher Block Chaining (CBC) mode of encryption can be used to prevent this type of modification attacks. Existential forgery is possible with inappropriate choice of a redundancy function R in modular squaring. For instance, in case of $R(m) = m$, an adversary can select any integer $a \in Z_{nCA}^*$ and squares it to get $m' = a^2 \bmod n$. Then, a is a valid signature for fabricated m' without the knowledge of private key. It is strongly recommended to use the redundancy function R as specified in ISO/IEC 9796 digital signature standard. Also, using Modified Rabin's Signature scheme [14] for modular squaring overcomes the problem of finding square root in signing process.

The security of proposed group key management scheme relies on random selection of personal secret key at user side and the cryptographic security of used hash function. SHA1 and MD5 hash algorithms are not theoretically secure. However, there is no successful attack on full round algorithms other than brute-force [2].

Proposed group key management mechanism provides the following security features:

(1) *Backward and forward secrecy*: Sibling of joined member generates a new personal secret key and modifies the keys of ancestors with consecutive hashing operations applied to new key in upward direction. New keys of the branch are encrypted by blinded keys of ancestor's siblings before transmission to group. Since all keys on the branch are modified, new member cannot access to any key used in the past. Evicted user does not have the knowledge of the blinded keys that its sibling holds. Since all ancestor keys are renewed by the sibling after eviction process, evicted user cannot access to future group encryption keys.

(2) *Free of collusions*: Previous keys are not used in derivation of new keys. So, any cooperation of evicted users cannot recover the future keys from the keys used in past.

3.5 Performance Analysis

The proposed key establishment protocols are simulated on highly integrated RISC processor, ARM920T, which is widely used microprocessor in wireless handheld devices and mobile phones. Simulated device has 133 MHz clock rate and uses 32-bit ARM instruction set. Practical and efficient elliptic curve cryptography library has been designed over $GF(p)$, where p is prime modulus.

Modular arithmetic module performs an improved version of Montgomery multiplication algorithm with modulus p for ECC operations and with 1024 bits modulus for modular squaring [9].

Table 3-1. Comparisons of ECC based key establishment protocols

Algorithm Comparison	Random Point Mult.	Fixed Point Mult.	Modular Multip.	Comm. Complexity
ECDHE	2	3	-	1796 bits
ECMQV implicit	2	1.5	-	1478 bits
Aydos-Y-K	1	2	-	1730 bits
Proposed	1	-	1	1632 bits

The scalable point arithmetic module is implemented only in 28 KB of code size for 160 bits key-length ECC curve. Jacobian coordinates have been used to represent points in the most time-efficient way. AES secret key encryption algorithm is selected with 128 bits key length, 128 bits block length and CBC (Cipher Block Chaining) mode of operation. SHA-1 hash algorithm is selected for hashing operations, which is implemented as standardized in FIPS 180-1.

Key establishment with *KSS* members repeats d-1 times in user join, where d shows the depth of tree after join. Comparison results are given in **Table 3-1** for several ECC based key establishment protocols. The "Proposed" refers to the protocol for authenticated key establishment with *KSS* members, since this protocol is the most computationally intensive part of join process. **Table 3-1** also compares the communication complexities in terms of total message size.

Real-time execution of key establishment with *KSS* members has three exchanged messages. Each user performs four hashing operations, one elliptic curve random point multiplication, one modular multiplication (squaring) with 1024 bits modulus, one symmetric key operation (encryption or decryption), one xor operation, one MAC computation and one random number generation. Random point multiplication takes 45.1 ms for the 160 bits ECC curve and squaring with 1024 bits modulus takes only 1.2 ms. Hashing and symmetric encryption/decryption are much less time consuming operations than expensive elliptic curve multiplications. Therefore, timings for these operations are ignorable. Efficiency of the proposed protocol lies in reduced number of expensive point multiplications.

As seen in **Table 3-1**, proposed protocol has comparable bandwidth efficiency and requires less computation than other ECC based key establishment protocols. Communication complexity is calculated as choosing k = 128, 64 bits *Id*, 32 bits time-stamp (t) and 1024 bits modulus.

Table 3-2. Comparisons of the costs for user join and leave operations

		Message Size		Required Computation	
		Unicast	Multicast	Key Distribution Center	Members
Proposed	Join	$d(k + is) + k$	$2(d - 1)(k + is)$	-	$G + d(2E + 2h + X + D) + h - E$
	Leave	-	$2(d - 1)(k + is)$	-	$G + (d - 1)(E + X + h) + dh$
OFT	Join	$(d + 3)k + is$	$(d + 1)k$	$G + (d + 1)h + d(X + 3E)$	$(d + 3)D + 2d(h + X)$
	Leave	-	$is + (d + 1)k$	$d(E + h + X)$	$D + d(h + X)$
ELK	Join	$(d + 1)k + is$	-	$G + (4n - 2)E + (d + 3)E$	$(d + 1)D + 2dE + 2E$
	Leave	-	$is + d(n_1 + n_2)$	$8dE$	$dD + 5dE$
DTKM	Join	$2dk$	$(d - 1)k$	-	$2d(PE + PD) + (d - 1)E + d(X + h)$
	Leave	$(d - 1)k$	$(d - 1)k$	-	$(d - 1)(PD + PE + E + X + h)$
CTKM	Join	Dk	$d.is$	(Group Man): $KS + E + dh$	$KS + D$
	Leave	$(2d - 1)k$	-	(Group Man): $G + (2d - 1)E$	-

Each user should contact to $d=\log_2 n$ member to complete the join process, where n refers to number of members. The performance results of the proposed group key management scheme and selected HBT based schemes are compared. Evaluation is based on the number of required unicast and multicast messages and computation overload for each join and eviction. **Table 3-2** presents the cost of re-keying for each join and leave process to the group in terms of total message size and computations required. The notation used throughout the tables is given in **Table 3-3**.

Table 3-2 shows that KDC employing schemes require smaller number of computations for members. However, these schemes transfer the major part of the computational load to KDCs. Although DTKM does not employ any KDC, group re-keying with public key cryptography makes DTKM slower than the proposed scheme.

The number of unicast messages is comparable to other schemes. However, sibling sends out additional d-1 multicast messages for each join and eviction. Thus, proposed scheme uses more bandwidth than other schemes for multicasting. Note that, $n_1 + n_2 = K$ is chosen for the ELK protocol to provide same level security with other schemes. Finally, every member in the proposed scheme holds $(2d + 1)$ keys, d indexes and d entries of $(Id \| P \| t)$ for the group key management.

Table 3-3. Notation

Symbol	Meaning
d	Depth of the tree
E	Secret key encryption operation
K	Size of secret key in bits
D	Secret key decryption operation
Is	Size of index in bits
PE	Public key encryption operation
n	Number of group members
PD	Public key decryption operation
G	Key generation
X	Xor operation
h	Hash function
KS	Key setup process between member and group manager

3.6 Conclusion

In this paper, a practical and efficient group key management framework is presented for resource constrained mobile ad hoc networks. The proposed HBT based model does not require any central key management authority or group manager, and achieves comparable performance with the models which use secret key cryptography and employ key distribution authority. Customized hybrid authenticated key establishment protocols are presented for more efficiency in membership modification.

According to the comparisons, proposed hybrid protocols achieve comparable results in terms of bandwidth efficiency and better results in terms of processing time.

3.7 References

[1] M. Aydos, T. Yanik and C. Koc. An high-speed ECC-based wireless authentication protocol on an ARM Microprocessor. *Proceedings of ACSAC'2000*, New Orleans, LA, December 2000.

[2] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Cryptographic hash functions: A survey. *Technical Report 95-09*, University of Wollongong, July 1995.

[3] S. Capkun, L. Buttyan, and J. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52-64, January-March, 2003.

[4] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner. Efficient security for large and dynamic multicast groups. *Proceedings of the Seventh Workshop on Enabling Technologies (WET ICE'98)*, IEEE Computer Society Press, 1998.

[5] Certicom Research, Standard for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 1.0, September 2000.

[6] L. R. Dondeti, S. Mukherjee, and A. Samal. A distributed group key management scheme for secure many-to-many communication. *Technical Report PINTL-TR-207-99*, Department of Computer Science, University of Maryland, 1999.

[7] O. M. Erdem. High-speed ECC based Kerberos authentication protocol for wireless application. *IEEE GLOBECOM'2003 Communication Security Symposium*, San Francisco, CA, December 2003.

[8] D. A. McGrew and A.T. Sherman. Key establishment in large dynamic groups using one-way function trees. *Technical Report No.0755*, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[9] P. L. Montgomery. Modular multiplication without trail division. Mathematics of Computation, Vol. 44, No. 170, pg. 519-521, 1985.

[10] A. Perrig, D. Song, and J. D. Tygar. ELK, a new protocol for efficient large-group key distribution. *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

[11] S. Rafaeli, L. Mathy, and D. Hutchison. EHBT: An efficient protocol for group key management. *Proceedings of the Third International COST264 Workshop (NGC 2001)*, Springer-Verlag, LNCS 2233, pages 159-171, London, UK, November 2001.

[12] SECG, Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, 2000. Available from http://www.secg. org/collateral/sec1.pdf.

[13] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A new approach to group key agreement. *Technical Report RZ 2984*, IBM Research, December 1997.

[14] H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, 26 (1980), 726-729.

[15] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6), November-December 1999.

CHAPTER 4

LIGHTWEIGHT KEY ESTABLISHMENT PROTOCOLS FOR SELF-ORGANIZING SENSOR NETWORKS

Özkan M. Erdem

School of Electrical Engineering and Computer Science

Oregon State University

Corvallis, OR 97331

To be submitted to IEEE 2nd Annual Communication Society Conference on Sensor and Ad Hoc Communications and Networks, SECON'05

Santa Clara, CA, USA, September 2005.

Abstract

Introducing mutual authentication and link encryption to the communication between sensor and data processing station requires more specialized approaches since sensors have constraints in battery power, communication and computational capabilities. Public key cryptography based schemes offer self-organized trust establishment. However, constraints in resources make fully public key operations not affordable on sensor networks. This paper presents three different authenticated key establishment protocols which combine public-key cryptography, secret-key cryptography and one-way hash functions. Objective of the proposed protocols is to avoid using expensive modular exponential and elliptic curve point multiplications. Sensor needs to make only modular or cyclic convolution multiplications, and expensive public key decryption operation is executed at the station side. Proposed schemes mutually authenticate the sensor and station before they initiate the secure data communication and establish distinct keys for every session. The proposed schemes are also implemented on Texas Instruments MSP430 microcontroller with small code size and achieved the least sensor processing time and the lowest battery power consumption in comparison with fully public key cryptography based protocols.

4.1 Introduction

Wireless sensor networks offer economical real-time data processing solutions to wide variety of applications in dynamic and complex environments. Deployment of inexpensive sensor networks became more feasible with continuous technological improvements in ad hoc networking protocols, embedded systems and low-cost wireless communication. Sensor node is a battery powered microprocessor integrated with sensor equipment and short-range radio communications. In near future, hundreds or thousands of sensor nodes are expected in every sensor networks. Potentially, sensor networks serve to both civilian and military applications, including operation control, surveillance, vehicle tracking, environment monitoring, health care and maintenance of manufacturing systems.

The privacy and security in wireless sensor networks become essential when the security of deployment area cannot be provided. Primarily, each sensor behaves as a potential point of several physical and logical attacks. Attackers can capture and reprogram individual sensor nodes. Due to wireless nature of sensor network, transmission between nodes can easily be monitored and private sensed data or control information can be disclosed. Finally, adversaries induce battery exhaustion, prevent the functionality of network or maliciously use the subset of nodes for illegal purposes.

Tamper resistant hardware solutions could make reprogram of captured node infeasible. However, sensor nodes are inexpensive, low powered devices with limited computational and communication resources. Production of tamper resistant and robust nodes may not be economically viable option. Therefore, one must rely on software based security solutions against possible attacks and assume that subset of nodes is compromised. Encryption of communication between nodes and node-to-

node authentication are crucial building blocks of sensor network security. Node-to-node authentication provides several advantages, such as initial link key establishment between nodes and revocation of malicious nodes. However, design of light-weight authentication and link key establishment protocols is a rich research field and stays as an open problem.

Key establishment schemes can be classified into three different categories. The trusted central authority scheme employs a single or multiple servers to establish the trust among nodes and to manage the keys, e.g. Kerberos [20]. Self-organizing sensor networks usually do not have trusted infrastructure, so this type of centralized security solution does not work [8].

In second type of scheme key information is distributed among nodes prior to forming the group. Pre-distribution of the keys can be managed in two basic ways. In first approach, each sensor carries pairwise secret link key table. Each key table has N-1 distinct pairwise secret keys, N is the total number of sensors. Each key on the table is shared by only two sensors and used to encrypt the traffic between these two sensors. This scheme is especially suitable for computationally weak sensors. However, bandwidth and memory efficiency significantly decreases when the number of sensors increases. Besides, link keys for the sensors which will join to the network in future cannot be pre-installed to other sensors. Alternatively, each sensor carries *group master secret key* and uses this key to derive distinct pairwise link keys [4]. This method is quite useful for large-scaled sensor networks. However, method is very insecure; compromise of one of the sensors can reveal the *master secret key*, and so the entire network can be compromised.

Public key cryptography based schemes offer self-organized trust establishment between sensors with the help of key certificates. Compromise of any device does not

pose a security risk to other sensors. However, constraints in computational capability, bandwidth and battery power make solely public key operations not affordable on sensor networks. On the other hand, security protocols based on hybrid cryptographic operations would be more respectful to resources of sensors and provide strong security and authentication features of public key cryptography.

4.1.1 Related Work

Although majority of research on security of wireless sensor networks emphasize the secure routing protocols, there are some proposals on authentication and key management issues.

Due to computationally heavy operations in public key cryptography, most of sensor network security protocols are based on secret-key cryptosystems. Perrig et al. addressed security issues in wireless sensor networks, and proposed SPINS security protocol which is solely based on secret-key cryptography [21]. SPINS employs an online trusted base station and sensors have master keys established with trusted station. However network needs to have more than one station in self-organizing dynamic environment. Then, a new problem arises in SPINS which is how to embed all future master keys into sensors. Du et al. proposed a key pre-distribution scheme, which substantially improves the resilience of wireless sensor networks [8]. In proposed scheme each node establishes pair-wise secret key with other node even if they are in different key spaces. When nodes have different key spaces, pair-wise secret key is transmitted from one node to another through *key-sharing graph.* However, using key-sharing graph exposes pair-wise secret keys to other sensors in group. So, this makes scheme not applicable for the networks which is deployed in a hostile environment and sensors are vulnerable to compromise.

Cam et al. proposed an energy-efficient security protocol for sensor networks. This scheme uses dynamically changing session keys and code-hopping techniques [5]. In [16], Jolly et al. presents a key management protocol which is based on Identity-Based Symmetric Keying (IBSK) scheme. Key pre-distribution problem is solved as embedding two keys to each sensor in pre-deployment phase. Sensor shares one key with a static gateway and shares another key is with the command node. However, this scheme cannot be useful when gateways are dynamically relocating from one sensor cluster to another cluster. Besides, scheme forces deployment of gateways in secured environment since compromise of a gateway compromises all sensor keys in the same cluster.

There are also recently proposed public-key cryptosystem based schemes in securing sensor networks. Wang and Chan proposed two key exchange protocols; the server-specific MAKEP and the linear MAKEP [28]. In these schemes, the client has to perform expensive modular exponential operations that are not viable for computationally constrained sensors.

Usage of ECC has been proposed several times in wireless key agreement protocols. ECC based protocols use smaller key lengths to provide strong security. Huang et al. introduced an authenticated key establishment protocol for self-organizing sensor networks [14]. They combined ECC with secret-key cryptography, namely Hybrid, and also presented MSR-combined version of their key establishment. While they focus on reducing the computational load in sensor side, sensor still needs to perform several expensive point multiplications which are not affordable. ECMQV protocol with implicit certificates [6], ECDSA authenticated key exchange protocol [2], the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) protocol [25] and Aydos-Yanik-Koc key agreement protocol [2] are other ECC based wireless key agreement schemes.

Similarly, all of ECC based schemes mentioned above require at least three expensive point multiplication operations at sensor side if entity and key authentication are provided.

4.1.2 Contributions

Dynamically relocating stations: While sensors stay static in the sensor network, data collecting stations can relocate dynamically and move from one sensor cluster to another cluster. Proposed schemes establish distinct keys for every session between any data collecting station and any sensor in the system as long as they are in the same cluster at that moment.

Mutual authentication: Sensor and data collecting station mutually authenticate each other before they initiate the secure data communication.

Lightweight operations on sensor side: Main objective is introducing public key cryptography to the sensor-station key establishment. However, proposed protocols avoid using expensive modular exponential or elliptic curve point multiplications. Sensor needs to make only large or small modular multiplications, secret-key encryption/decryption and hashing operations.

Power efficient key establishment: Maximization of life-time of sensor is possible only with using power efficient key establishment protocols. Although the strong security advantage of public key cryptography is taken, proposed schemes consume only 15-28 mW of battery power.

4.2 Background of Fast Public Key Cryptography

Similar to RSA public key encryption algorithm [24], Rabin's scheme [23] is based on the factorization problem of large modulus. Key encryption and signature verification processes in Rabin's scheme are extremely fast and this makes scheme suitable for low power and resource constrained devices. Rabin's scheme requires one modular multiplication for encryption in which ciphertext is $c = m \cdot (m + b) \bmod n$, for the message m. In other words, the encryption process requires only one squaring if b is fixed to 0. Protocols are constructed in such a way that sensor does not need to perform computationally expensive decryption operation. More details of the Rabin's encryption scheme can be found in [23, 29].

NtruEncrypt is a highly efficient and relatively new public key cryptosystem. Underlying Ntru is a hard mathematical problem of finding short and/or close vectors in a certain class of lattices or Ntru lattices [11, 12, 13]. Simplicity of its underlying arithmetic makes NtruEncrypt algorithm especially suitable for sensor networks and RFID tags.

Despite the shorter history in public literature, resistance of Ntru algorithms to cryptanalysis is claimed to be comparable to RSA and ECC algorithms.

Brief key generation, encryption and decryption steps for NtruEncrypt algorithm are presented in Appendix A. More details on NtruEncrypt can be found in [11].

4.3 Sensor Network Model

Sensor network can be formed in various different models according to the desired application. Envisage a military scenario such that sensor network is deployed in a hostile environment and sensors are used to track the movements of enemy. In the

proposed model, a sensor network consists of a large number of sensors, multiple data processing stations and command base. The hierarchical components of the network are defined as in the following;

Sensors: Sensors are smallest elements of the network and randomly deployed in a certain geographical location. Sensors are assumed to stay static. Each sensor determines its location by GPS system during bootstrapping and location information is recorded by command base and stations. Sensor provides one-way data transmission to the stations. Prior to data transmission it establishes the session key with the station and encrypts the data traffic.

Stations: Stations are intermediate elements between sensors and command base. They collect the data from sensors in the same cluster and forward to command base. Stations have higher computational capabilities and larger memory than sensors. Each station has the ability to establish a secure direct connection with other stations to make key or data exchanges. Different from other proposed models in [5, 9, 16], stations can relocate dynamically in proposed approach. This reflects the following situations in sensor network:

i. Sensors and stations are in hostile environment. Stations are static but open to compromise and compromised station needs to be evicted from the network. Another station is assigned by command base for the same task.

ii. Sensors and stations are in hostile environment. Mobile station collects the data off-line for the specified time interval and moves to another sensor cluster.

Command base: The command base, which is called as *CB* throughout this paper, behaves as central management and certification authority. *CB* is assumed to be locating in secure environment. Command base manages the clustering the network, assigning/evicting sensors and assigning/evicting the stations, as well. Various

security protocols can be used to encrypt the traffic between the command base and stations.

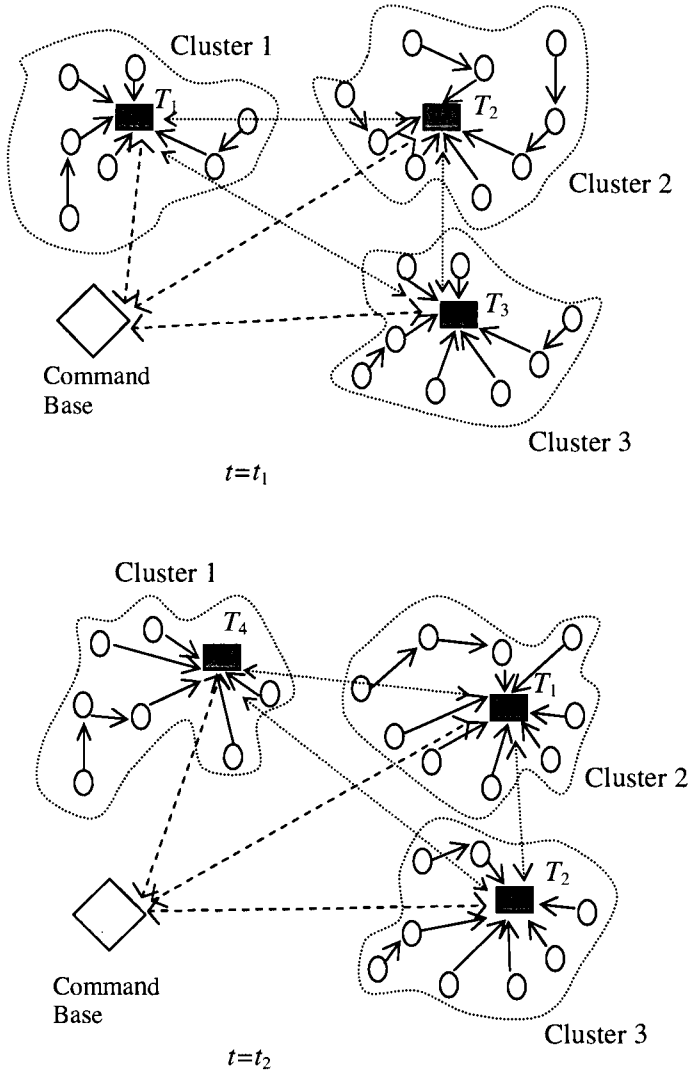

Figure 4-1. Sensor network architecture with mobile stations

Figure 4-1 illustrates the possible sensor network scenario in which sensors continuously establish connections with several stations. In $t=t_1$ moment, scenario uses T_1 for Cluster 1, T_2 for Cluster 2 and T_3 for Cluster 3. However in $t=t_2$ moment

the structure of network changes and T_3 becomes unavailable. T_4 controls the Cluster 1, T_1 relocates and collects data from Cluster 2 sensors, T_2 relocates and collects data from Cluster 3.

Next chapter focuses on certification of stations and session key establishment models between station and sensor.

4.4 Authenticated Key Establishment Protocols

In proposed key establishment protocols, sensors can communicate only with the stations which have valid certificate issued by command base. In other words, sensors authenticate the station before transmission of sensed data and establish a session key to encrypt the data. In this chapter, three different authenticated key establishment protocols are given. First proposed protocol is based on modular multiplication which loads the majority of work to the station side. Second proposed protocol uses simplified version of previous protocol. Third protocol improves the computational efficiency by employing NtruEncrypt encryption algorithm [10].

Sensors prove their identities with the knowledge of their private response generation key which is common method for both protocols. Every sensor carries a unique secret key which is embedded to sensor prior to placed in the network. Sensor also shares this response generation key with the command base.

Challenge-Response list is proposed for each station. Challenge-Response list, in short CR list, uniquely identifies the associated *response* value for each possible *challenge* value which must be expected from the sensor to authenticate the sensor to the station. Distinct CR lists are provided to each station when they obtain their public key certificates from *CB*. **Table 4-1** illustrates an example CR list which is provided to one of the stations in the network by *CB*. First, *CB* determines the sensors with which

the station has the authorization to make data transfer. Then, *CB* generates a new CR list specific to the station and inserts the identities of sensors, random challenge (*Ch*) and the response (*Rs*) values.

Table 4-1. Example Challenge-Response list for the station

Sensor Id	Challenge (Ch)	Response (Rs)
S00001	Ch_{S00001}	Rs_{S00001}
S00017	Ch_{S00017}	Rs_{S00017}
S04521	Ch_{S04521}	Rs_{S04521}

4.4.1 Modular Multiplication Based Key Establishment

Modular multiplication based authenticated key establishment protocol comprise of Rabin's scheme to encrypt the sensor's contribution to session key and put the computational burden in station side. Implicit certification is used to verify the identity and public key of station. As a definition, public-key certificate bonds the identity of the device to the public key information and digitally signed by the *CB*. *CB* stores the certificate verification public key of station. Implicit certificate in modular multiplication based scheme is defined as encrypted value of concatenation of identity information, public key constructor and expiration date of the certificate. Encryption is performed by command base. Verification of implicitly certified key consumes less bandwidth and computational power in comparison with explicit certification.

Pre-assumption: Command base (*CB*) selects m-bit random prime numbers f and g, which provide the condition of $f \equiv 3 \pmod 8$ and $g \equiv 7 \pmod 8$. *CB*'s public key modulus for the Modified Rabin's Signature algorithm is $n_{CB} = fg$ and private key is $d_{CB} = (n_{CB} - f - g + 5)/8$. *CB* assigns 32-bit unique identification number *Id* for each sensor and station in the network. This identity string can include the name of device

in the network and the location information. Public key of *CB* is published to stations and stored in sensor devices.

All certification protocols introduced in this paper are one time protocols and performed off-line. Transmission of messages is executed in secure way. The notation used in this chapter is specified in Table 4-2.

Table 4-2. Notation

Symbol	Meaning
CERTreq	Request message for certification
CERTres	Response message for certification
KeyAgReq	Key agreement request
KeyAgRes	Key agreement response
KeyConfReq	Key confirmation request
KeyConfRes	Key confirmation response
Id_i	Network identification information of sensor or station i
$Cert_T$	Certificate of station T
$H(x)$	One-way hash function output of x
$R(m)$	Redundancy function applied to m
$R^{-1}(m)$	Inverse redundancy function applied to m
$E[K, M]$	Secret-key encryption (CBC mode) of the message M under the key K
$D[K, M]$	Secret-key decryption (CBC mode) of the message M under the key K
$MAC[K, M]$	Message authentication code of the message M under the key K
CR_T	Challenge-Response table for station T
CRS_T	Challenge-Response-Signature table for station T
$\|$	Concatenation
$msb_x(M)$	The most significant x-bit of M
$lsb_x(M)$	The least significant x-bit of M

The detailed protocol steps are shown in **Figure 4-2** and given in the following:

i. Station T selects n-bit random numbers p and q and computes the public key modulus $n_T = p.q$. Certification request (*CERTreq*) message is sent to *CB*. Message contains the Id_T and the public key n_T. Station T securely stores the private key (p, q).

ii. CB checks the validity of Id_T and verifies the status of T in the network. Following verification, *CB* generates a time-stamp value t_T and computes $y_T = H(Id_T) \| H(t_T)$. Then, *CB* applies previously identified redundancy function R to the exor of y_T and the public key n_T. Redundancy addition is crucial in Rabin's scheme to prevent existential forgery attacks.

Station (T)	***Command Base (CB)***
Select n bit random primes p and q	
Compute $n_T = p.q$	
T's public key is n_T and private key is (p, q)	
Send ***CERTreq***: $\{Id_T, n_T\}$ $\longrightarrow$	
	Receive ***CERTreq***
	Check validity of Id_T.
	If Id_T is not valid, then abort.
	Generate time-stamp t_T
	Compute $y_T = H(Id_T) \| H(t_T)$
	Compute $P_T = (R(n_T \oplus y_T))^{d_{CB}} \bmod n_{CB}$
	Prepare the Challenge-Response list CR_T.
	$Cert_T = \{Id_T, t_T, P_T\}$
	Send *CERTres*: $Cert_T$, CR_T.
$\longleftarrow$	
Receive ***CERTres***	
Compute $P_T' = P_T^{\,2} \bmod n_{CB}$.	
Compute $y_T' = H(Id_T) \| H(t_T)$	
Compute $n_T' = R^{-1}(P_T') \oplus y_T'$	
If n_T' n_T, then abort.	
Store $Cert_T$ and CR_T.	

Figure 4-2. Certificate generation for the station T

Thereafter, *CB* encrypts the output of $R(n_T \oplus y_T)$ with its private key d_{CB} and gets the public key constructor P_T. Certificate of T is simply concatenation of Id_T, t_T and P_T.

iii. *CB* identifies the sensors that are in the list of T to collect data, generates unique i-bit challenge Ch value for each sensor, and then computes associated response Rs for each Ch by using response generation key e of each sensor; $Rs = H\ (Ch \oplus e)$. Certification Response (*CERTres*) message containing certificate and Challenge-Response list (CR_T) is sent back to the station T.

iv. Having certification response message, T computes $y_T = H\ (Id_T) \parallel H\ (t_T)$ and decrypts P_T by squaring modulo n_{CB} $(P_T^{'} = P_T^2 \bmod n_{CB})$. After applying reverse redundancy function R^{-1} to $P_T^{'}$, T accepts the received implicitly certified public key only if the result of $(R^{-1}(P_T^{'}) \oplus y_T)$ is equal to its own public key n_T. Otherwise T rejects the certificate and aborts the protocol.

Authenticated session key establishment protocol is executed between station and sensor for every data transmission session. The details of protocol are given in below and steps are illustrated in **Figure 4-3**.

i. To initiate the key establishment protocol, station T picks challenge value Ch_S for sensor S from CR_T and sends key agreement request (*KeyAgReq*) message which contains $Cert_T$ and Ch_S.

ii. Having received key agreement request, sensor S computes public key of T. To achieve this, S takes square of P_T and applies inverse redundancy function R^{-1} to the result. Public modulus n_T is calculated as

$$n_T = R^{-1}(P_T^{\ 2} \bmod n_{CB}) \oplus (H\ (Id_T) \parallel H\ (t_T)).$$

iii. S computes associated response value to Ch_S by using its response generation key e_S; $z_S = H\ (Ch_S \oplus e_S)$. The response z_S should be sent back to T to prove identity. However, message has to be encrypted to prevent impersonation attacks. S also chooses k-bit random number r_S to be used as sensor's contribution to session and MAC keys. S encrypts the message of $(x \parallel r_S \parallel z_S)$ by simply squaring it

modulo n_T prior to sending key agreement response *KeyAgRes*. Here x shows the random binary string used in padding.

iv. Station T takes square root of received number q in modulo n_T to decrypt. Using Modified Rabin's algorithm guarantees to find square root m successfully. Then T removes the padding and picks the least significant $k + 160$ bit of m. First k-bit of selected part is taken as sensor's contribution r_S and the last 160 bit z_S is compared with Rs_S. If comparison fails, then T does not authenticate the sensor and terminates the protocol. Otherwise, T generates k-bit random number r_T as its own contribution to session key SK and message authentication code (MAC) key MK.

v. Key confirmation request message *KeyConfReq* is sent back to S encrypted with r_S (*KeyConfReq*: $c = E\ [r_S,\ \ r_T \,||\, Id_T]$). Sensor decrypts the received c and obtains $r_T \,||\, Id_T$. Verification of existing Id_T in the decrypted message confirms that station has received and decrypted r_S successfully. If Id_T is not correct, then S terminates the protocol.

vi. Sensor S and station T compute the 128-bit SK and 128-bit MK with the following formulas:

$$SK = msb_{128}\,(H(msb_{\lfloor k/2 \rfloor}\,(r_S)\,||\, msb_{\lfloor k/2 \rfloor}\,(r_T)\,||\, msb_{32}\,(z_S))\,),$$

$$MK = msb_{128}\,(H\,(lsb_{\lfloor k/2 \rfloor}\,(r_S)\,||\, lsb_{\lfloor k/2 \rfloor}\,(r_T)\,||\, lsb_{32}\,(z_S))\,),$$

where $lsb_x\ (y)$ denotes the least significant x bit of y, and $msb_x\ (y)$ denotes the most significant x bit of y.

Station (T)	***Sensor (S)***
Pick Ch_S for the sensor S from the list CR_T. Send ***KeyAgReq***: $Cert_T$, Ch_S	
$\longrightarrow$	
	Receive ***KeyAgReq*** If t_T is not valid, then abort. Compute $P_T' = P_T^{\,2} \bmod n_{CB}$ Compute $y_T' = H(Id_T) \,\|\|\, H(t_T)$ Compute $n_T = R^{-1}(P_T') \oplus y_T'$ Compute $z_S = H(Ch_S \oplus e_S)$ Choose k-bit random number r_S. Compute $q = (x \,\|\, r_S \,\|\, z_S)^2 \bmod n_T$ Send ***KeyAgRes***: q
$\longleftarrow$	
Receive ***KeyAgRes*** Compute $m = \sqrt{q} \bmod n_T$ The least significant $(k + 160)$ bits of $m = r_S \,\|\, z_S$ If z_S Rs_S, then abort. Choose k-bit random number r_T. ***KeyConfReq***: $c = E[r_S, r_T \,\|\, Id_T]$ Send ***KeyConfReq***	
$\longrightarrow$	Receive ***KeyConfReq*** $c' = D[r_S, c] = r_T \,\|\, Id_T$ If Id_T is not valid, then abort.
$SK = msb_{128}(H(msb_{\lfloor k/2 \rfloor}(r_S) \,\|\, msb_{\lfloor k/2 \rfloor}(r_T) \,\|\, msb_{32}(z_S)))$ $MK = msb_{128}(H(lsb_{\lfloor k/2 \rfloor}(r_S) \,\|\, lsb_{\lfloor k/2 \rfloor}(r_T) \,\|\, lsb_{32}(z_S)))$	
	KeyConfRes: $w = MAC[MK, Id_S]$ Send ***KeyConfRes***
$\longleftarrow$	
Receive ***KeyConfRes*** Compute $w' = MAC[MK, Id_S]$ If w w', then abort.	

Figure 4-3. Modular multiplication based key establishment protocol

vii. Last step in key establishment protocol is confirming the keys are established correctly. The method is having sensor S to send back generated MAC code of Id_S. Station gets this message and compares with the output which it gets using MK. If MAC codes match, then session key establishment protocol is completed. Otherwise, T repeats the protocol.

4.4.2 Lightweight Key Establishment Protocols

In this chapter computational efficiency of modular multiplication based protocol is improved and two different lightweight key establishment protocols are proposed. Similarly, these protocols puts the cryptographic burden on station and less workload on sensor. First scheme uses modular multiplication for encryption of sensor's contribution to session keys whereas in the second scheme NtruEncrypt encryption operation is replaced with modular multiplication. Unlike the proposed protocol in previous chapter, sensor and station authenticate each other using only hash functions and secret-key encryption algorithms.

Pre-assumption: *CB* assigns 32-bit unique identification number *Id* for each sensor and station in the network. Identity string can include the name of device in the network and the location information. In both schemes, sensors carry the same response generation i-bit response generation key e_S and a unique i-bit MAC verification key u_S. This key is assigned by command base to every sensor and used in verification of the public key of station. The concept of CRS (Challenge-Response-Signature) list is introduced here. This list is replaced by CR list on the station and it contains the *Ch, Rs* and *Sg* values. Command base generates the MAC value of station's public key using associated sensor's MAC verification key u_S and assigns the

result to Sg. Thus, when the station is certified it gets different Sg value for every sensor in CRS list. Example CRS list is shown in **Table 4-3**.

Table 4-3. Example Challenge-Response-Signature list for the station

Sensor Id	Challenge (Ch)	Response (Rs)	Signature (Sg)
S00001	Ch_{S00001}	Rs_{S00001}	Sg_{S00001}
S00017	Ch_{S00017}	Rs_{S00017}	Sg_{S00017}
S04521	Ch_{S04521}	Rs_{S04521}	Sg_{S04521}

4.4.2.1 Modular Multiplication-Light Scheme

Two-way certification protocol steps between station and sensor for Modular multiplication-Light scheme are given in the following and illustrated in **Figure 4-4**:

i. Similar to multiplication based protocol, station T computes $n_T = p.q$ and sends Certification request ($CERTreq$) message to CB.

ii. Following verification of Id_T, CB generates a time-stamp value t_T and $Cert_T = (Id_T \| n_T \| t_T)$, identifies the sensors that are in the list of T to collect data, generates unique i-bit challenge Ch value for each sensor and finally computes associated response Rs for each Ch by using response generation key e of each sensor; $Rs = H(Ch \oplus e)$. Next, CB computes MAC values of $Cert_T = (Id_T \| n_T \| t_T)$ using one of the sensor's u_S for each time ($Sg_S = MAC\ [u_S, Cert_T]$). Challenge-Response-Signature list for this station (CRS_T) is generated and sent along with $Cert_T$ back to the station.

iii. T receives $CERTres$ message, stores the $Cert_T$ and CRS_T.

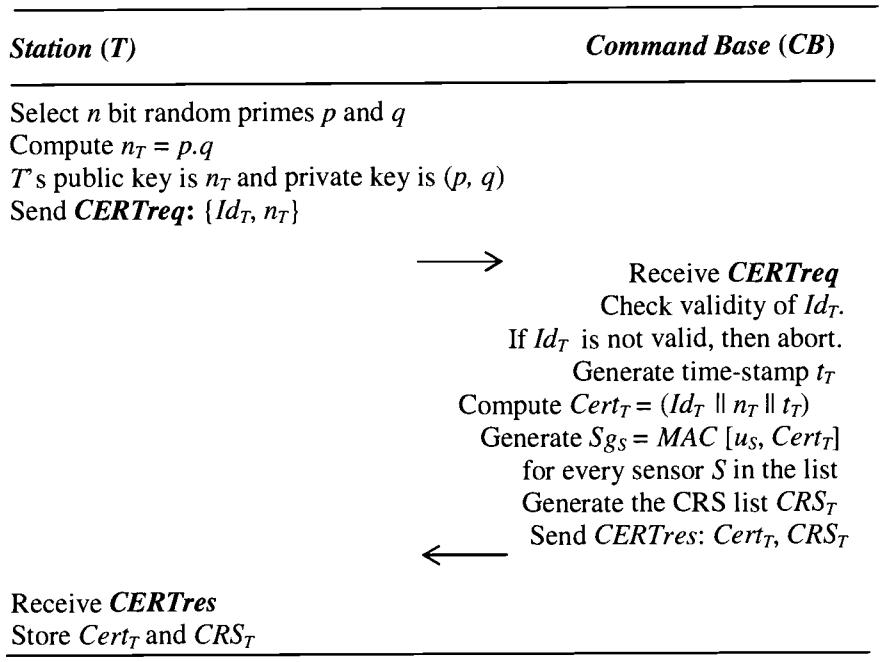

Figure 4-4. Certificate generation for the station ***T***

Station and sensor execute the authenticated session key establishment protocol before every data transmission. The details of protocol are given in below and shown in **Figure 4-5**.

i. Station *T* initiates the protocol, picks the challenge Ch_S and Sg_S for sensor *S* from CRS_T and sends key agreement request (*KeyAgReq*) message which contains $Cert_T$, Ch_S, and Sg_S.

ii. Sensor *S* receives *KeyAgReq*, validates the time-stamp t_T and computes the MAC of public key certificate $Cert_T$ using secret MAC verification key u_S. If the MAC result is equal to received Sg_S then sensor completes authenticating the station

and guarantees that MAC value has been generated by command base specifically for sensor S. Otherwise, S terminates the protocol.

Station (T)	***Sensor (S)***
Pick Ch_S and Sg_S for the sensor S from the list CRS_T.	
Send ***KeyAgReq***: $Cert_T$, Ch_S, Sg_S $\longrightarrow$	
	Receive ***KeyAgReq***
	If t_T is not valid, then abort.
	Compute $Sg_S' = MAC\,[u_S, Cert_T]$
	If $Sg_S \neq Sg_S'$, then abort.
	Compute $z_S = H\,(Ch_S \oplus e_S)$
	Choose k-bit random number r_S.
	Compute $q = (x \,\|\| \, r_S \,\|\| \, z_S)^2 \bmod n_T$
	Send ***KeyAgRes***: q
$\longleftarrow$	
Receive ***KeyAgRes***	
Compute $m = \sqrt{q} \bmod n_T$	
The least significant $(k + 160)$ bits of $m = r_S \,\|\| \, z_S$	
If $z_S \neq Rs_S$, then abort.	
Choose k-bit random number r_T.	
KeyConfReq: $c = E\,[r_S, \; r_T \,\|\| \, Id_T]$	
Send ***KeyConfReq*** $\longrightarrow$	
	Receive ***KeyConfReq***
	$c' = D\,[r_S, c] = r_T \,\|\| \, Id_T$
	If Id_T is not valid, then abort.
$SK = msb_{128}\,(H\,(msb_{\lfloor k/2 \rfloor}\,(r_S) \,\|\| \, msb_{\lfloor k/2 \rfloor}\,(r_T) \,\|\| \, msb_{32}\,(z_S)))$ $MK = msb_{128}\,(H\,(lsb_{\lfloor k/2 \rfloor}\,(r_S) \,\|\| \, lsb_{\lfloor k/2 \rfloor}\,(r_T) \,\|\| \, lsb_{32}\,(z_S)))$	
	KeyConfRes: $w = MAC\,[MK, Id_S]$
	Send ***KeyConfRes***
$\longleftarrow$	
Receive ***KeyConfRes***	
Compute $w' = MAC\,[MK, Id_S]$	
If $w \neq w'$, then abort.	

Figure 4-5. Modular multiplication-Light key establishment

The rest of the protocol is completed as executing the steps *iii*, *iv*, *v*, *vi*, and *vii* in modular multiplication based authenticated key establishment protocol.

4.4.2.2 NtruEncrypt-Light Scheme

As all proposed schemes NtruEncrypt-Light scheme has also two-way certification protocol which is securely executed one-time between station and command base. One major difference of the scheme is that stations use public-private key pair for NtruEncrypt algorithm instead of using key pair for Rabin's algorithm. The brief certification protocol is given in the following and illustrated in

Figure 4-6:

i. Station T chooses random polynomials F and g from the ring R; both polynomials should have small coefficients, then T computes the private key $f = 1 + pF$, and inverse of $f, f^{-1} \pmod{q}$. Encryption public key of station T is coefficients of $h \equiv g \otimes f^{-1} \pmod{q}$, which is denoted as h_T. Certification request (*CERTreq*) message is generated by T and sent to *CB*; message contains identity of T, Id_T, and public key h_T.

ii. CB checks the validity of Id_T, verifies the status of T and generates a time-stamp value t_T and $Cert_T = (Id_T \| h_T \| t_T)$. Similar to Modular multiplication-Light scheme *CB* identifies the sensors that are in the list of T, generates unique i-bit challenge Ch value for each sensor and computes associated response Rs for each Ch; $Rs = H(Ch \oplus e)$. Later, *CB* computes MAC values of $Cert_T = (Id_T \| h_T \| t_T)$ with the very same method as introduced in Modular multiplication-Light scheme. Challenge-Response-Signature list for this station (CRS_T) is generated and sent along with $Cert_T$ back to the station.

iii. T receives *CERTres* message, stores the $Cert_T$ and CRS_T.

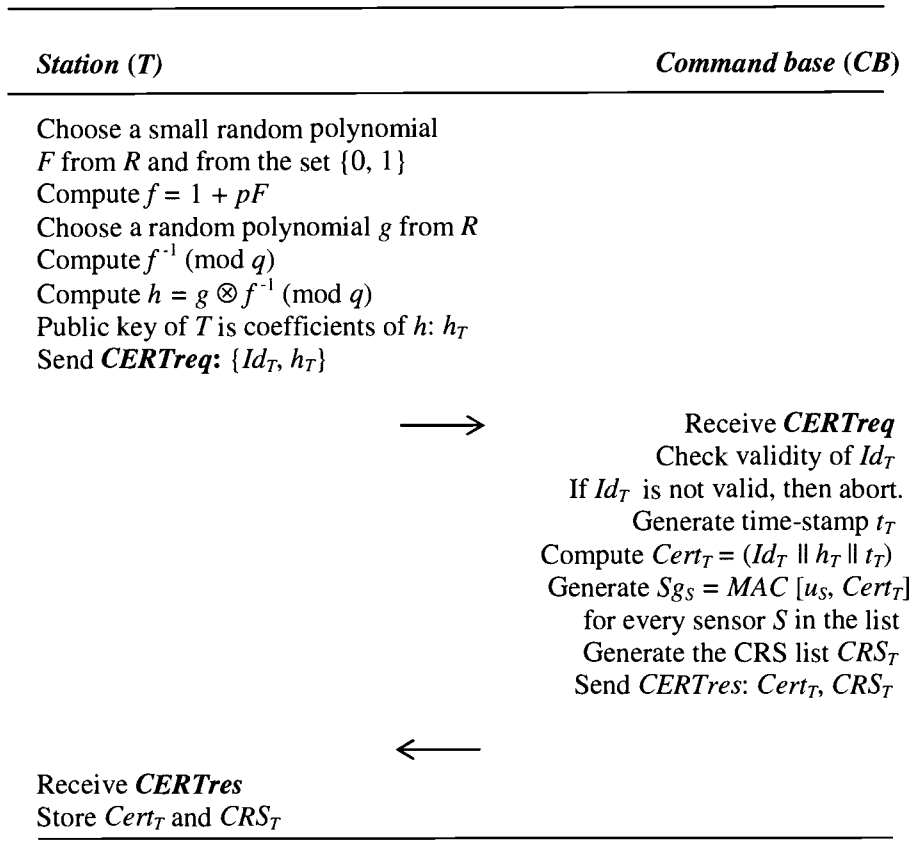

Figure 4-6. Certificate generation for the station T

Station and sensor authenticate each other and establish the session key prior to the every data communication as in other proposed schemes. Sensor verifies the public key certificate of station and sends its contribution to the session keys. Similarly station completes authentication of the sensor by sending pre-set challenge value and expecting to get correct response. Protocol steps are shown in **Figure 4-7** and the details are given in the following:

i. Station T sends key agreement request (*KeyAgReq*) message which consists of $Cert_T$, Ch_S and MAC value of $Cert_T$; Sg_S.

ii. Sensor S receives *KeyAgReq* and validates the time-stamp t_T. Protocol continues only if t_T is valid. Then, S computes the MAC of $Cert_T$ by using its secret MAC verification key u_S. If the MAC result is equal to received Sg_S then sensor completes authenticating the station. Otherwise, S terminates the protocol. Calculation of response $z_S = H(Ch_S \oplus e_S)$ is very same process as introduced in other schemes. S also selects k-bit random r_S, which is used as a contribution to session keys. Selected binary string is concatenated to the response z_S and result is padded with random x. Final text is assumed as plaintext m and encrypted with the following formula:

$$c = (p.\Phi \otimes h_T) + m \pmod{q}$$

As seen above, sensor makes one star multiplication to encrypt m, where Φ is a random polynomial from R. Ciphertext is sent to T as key agreement response (*KeyAgRes*).

iii. Station T decrypts c by making two consecutive star multiplications. Computation of $m' = (c \otimes f \pmod{q}) \pmod{p}$ gives $m \otimes f \pmod{p}$, and to extract plaintext T multiplies the result with f^{-1} and takes residue in modulo p. The least significant $k + 160$ bit of m is picked. Sensor's contribution r_S is the first k-bit of selected part, and the last 160 bit z_S is compared with Rs_S. If they are not equal, then T aborts the protocol. If equality is satisfied, T generates k-bit random r_T as its own contribution to create session key SK and message authentication code (MAC) key MK.

The rest of the protocol is completed as executing the steps *v*, *vi*, and *vii* in modular multiplication based authenticated key establishment protocol.

Station (T)	***Sensor (S)***
Pick Ch_S and Sg_S for the sensor S from the list CRS_T.	
Send ***KeyAgReq***: $Cert_T$, Ch_S, Sg_S	
$\longrightarrow$	Receive ***KeyAgReq***
	If t_T is not valid, then abort.
	Compute $Sg_S' = MAC\ [u_S, Cert_T]$
	If $Sg_S \neq Sg_S'$, then abort.
	Compute $z_S = H\ (Ch_S \oplus e_S)$
	Choose k-bit random number r_S.
	$m = (x \mid\mid r_S \mid\mid z_S)$
	Choose a random polynomial Φ from R
	Compute $c = (p.\Phi \otimes h_T) + m \pmod q$
	Send ***KeyAgRes***: c
$\longleftarrow$	
Receive ***KeyAgRes***	
Compute $m' = (c \otimes f \pmod q) \pmod p$	
Compute $m = m' \otimes f^{-1} \pmod p$	
The least significant $(k + 160)$ bits of $m = r_S \mid\mid z_S$	
If $z_S \neq Rs_S$, then abort.	
Choose k-bit random number r_T.	
KeyConfReq: $w = E\ [r_S,\ r_T \mid\mid Id_T]$	
Send ***KeyConfReq***	
$\longrightarrow$	
	Receive ***KeyConfReq***
	$w' = D\ [r_S, w] = r_T \mid\mid Id_T$
	If Id_T is not valid, then abort.
$SK = msb_{128}\ (H\ (msb_{\lfloor k/2 \rfloor}\ (r_S) \mid\mid msb_{\lfloor k/2 \rfloor}\ (r_T) \mid\mid msb_{32}\ (z_S)))$	
$MK = msb_{128}\ (H\ (lsb_{\lfloor k/2 \rfloor}\ (r_S) \mid\mid lsb_{\lfloor k/2 \rfloor}\ (r_T) \mid\mid lsb_{32}\ (z_S)))$	
	KeyConfRes: $b = MAC\ [MK, Id_S]$
	Send ***KeyConfRes***
$\longleftarrow$	
Receive ***KeyConfRes***	
Compute $b' = MAC\ [MK, Id_S]$	
If $b \neq b'$, then abort.	

Figure 4-7. NtruEncrypt-Light key establishment

4.5 Network Operations

4.5.1 Sensor Addition/Eviction

Prior to deployment of new sensor, public key of command base and private response generation key of the sensor are embedded in the sensor. This enables establishment of secure communication between sensor and stations.

Due to vulnerable nature of sensor networks, sensors can be out of battery, out of network access or compromised. In proposed model the responsibility of reporting these situations are given to the stations. This is because station is the closest entity to the sensor and has direct communication link with command base in particular geographical area. Since the station has the list of the sensors in that area it identifies unreachable sensor and reports sensor's identity to the command base. Command base automatically removes the sensor from the communication list and adds it to the revoked sensors list.

4.5.2 Station Addition/Relocation/Eviction

Stations can be added anytime to the network as it is required. Command base registers the new station to the system, certifies its public key and gives the location information in which station collects data. Every relocated station receives the new *CR* list from command base for the new geographic area. In case of lack in communication between relocating station and command base, station makes multi-hopping and use other stations to establish a path to the command base. Principally, command base does not allow two or more stations to collect data from the same sensor cluster unless each station contacts to different sensors in the cluster. If duty of the station requires obtaining data from other clusters to process its own data, then

station establishes a session key with each of other necessary stations by using its certificate and transfers the data securely.

Although stations are less vulnerable than sensors, they can be evicted from the network by command base in case of unavaliability or malfunctioning. Command base can use intrusion detection systems to detect compromised stations. New station is assigned instead of evicted one by command base, if necessary.

4.6 Security Analysis

Public key cryptography is introduced to provide key and entity authentication for the station in modular multiplication based key establishment scheme. Authentication of station is accomplished by using implicit certificate which binds the public key of device to the unique identity. Note that, public key n_T can be calculated only by using authentic public modulus n_{CB} so that man-in-the middle type attacks are mostly prevented. In Light protocols public key certificate of station is authenticated by each individual sensor using MAC verification key u_S. Compromise of any sensor does not pose a security risk to the other sensors. Because command base provides different MAC value to every station for the same public key certificate.

Since security also relies on secret-key cryptography, encryption algorithm must be properly implemented. Encrypting (r_T || Id_T) with ECB mode at *KeyConfReq* can lead modification attacks. Adversary can concatenate fabricated value of E [r_S, Id_T] to the true value of E [r_S, r_T] and send it to S as if it is originally coming from T. Cipher Block Chaining (CBC) mode of encryption can be used to prevent this type of modification attacks. Existential forgery is possible with inappropriate choice of a redundancy function R in modular squaring. For example, for $R(m) = m$ an adversary can select any integer $a \in Z_n^*$ and squares it to get $m' = a^2 \bmod n$. Then, a is a valid

signature for fabricated m' without the knowledge of private key. Using the redundancy function R as specified in ISO/IEC 9796 digital signature standard is strongly recommended [15]. Another problem can occur with finding square root in signing process. Using Modified Rabin's scheme [29] for modular squaring overcomes the difficulty of finding square root.

Challenge-response based authentication guarantees that sensor has the correct and valid e_S to prove its identity. Only one sensor in the network can response correctly to the challenge sent by station. Exor function combines the bit characteristics of e_S with Ch_S and one-way hash algorithm produces the digest value so that it is not feasible to guess e_S from the digest value. Although, SHA1 and MD5 hash algorithms are not theoretically secure, there is no successful attack on full round algorithms other than brute-force [3]. For this reason, one-way hash functions are accepted computationally secure. Also i=128 is used as the bit length of e_S, which is large enough to prevent brute force attacks to guess e_S for the immediate future.

Due to vulnerable nature of sensors compromise of subset of sensors in an unfriendly environment is tolerable. Backward secrecy is preserved if the sensor is compromised and private response generation key e_S is revealed. Attacker cannot recover the previously used link and MAC keys from e_S. Security of previously used keys also depend on the length of used r_S and padding string x. The minimum bit length of r_S and r_T should be k = 128 which is also important issue for generating distinct session link and MAC keys.

Backward secrecy in station-sensor communications can fail when the station is compromised. Since *KeyAgRes* message is encrypted with the station's public key, adversary can decrypt the earlier messages with revealed private key and can obtain r_S and r_T to find the session keys. However, forward secrecy for the sensors is still

preserved in this case. Although adversary obtains z_S for a specific sensor, it cannot be used to impersonate the sensor. Because command base sends unique CR list to every station so that sensor's response z_S is essentially different for every station. Backward secrecy in case of compromise of station may be possible by certification of sensor and sending *KeyAgReq* message encrypted with public key of sensor. This means sensor needs to do expensive public key decryption operation, which is against our purpose to minimize the workload of sensor.

After receiving *KeyAgReq* message sensor validates the certificate and verifies that station is true owner of the message. *KeyAgRes* message verifies the possession of the private sensor key to the station. Key confirmation is implicitly provided by added *KeyConfReq* and *KeyConfRes* messages. The final established keys are derived by equal contribution from each party. Thus, no single party has the entire control on derivation of session keys.

In generation of *SK* and *MK* session keys, random r_S, random r_T and digest value z_S are used. Since z_S cannot be counted as random number, only 32 bits of it is used in key generation while usage of r_S and r_T is maximized. Most significant bits of numbers are used to generate *SK* and the least significant bits of numbers are used to generate *MK*. Thus, we obtain totally different *SK* and *MK* in every session. Additionally, using hash function uniformly distributes the bit characteristics of random numbers into the digest value which is used as final *SK* or *MK*.

4.7 Performance Evaluation

4.7.1 Hardware Specifications

The proposed protocols are implemented and measurement results are obtained on MSP430 ultra low power microcontroller [26, 27] with 10 MHz clock rate which is specifically designed for sensor systems by Texas Instruments Incorporated. The architecture, combined with five low power modes is optimized to achieve extended battery life in portable measurement applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that attribute to maximum code efficiency. Selected device has flash memory, 1 KB RAM and consumes low power (400 μA at 1 MHz) in active mode. Integrated hardware multiplier performs a multiplication operation of two 16 bits operands only in 8 instruction cycles which is especially suitable for the proposed modular multiplication based protocol.

4.7.2 Cryptographic Software Modules

Implemented cryptographic software in C language contains modular arithmetic module, cyclic convolution product (star multiplication) module, secret-key encryption/decryption and hashing modules. We assume that we have a good source of random bits available for generation of random numbers r_S, r_T and random Φ polynomial. Aspect of random number generation is outside of scope of this paper.

Practical and efficient modular arithmetic module performs modular addition and improved version of Montgomery multiplication algorithm for modulus p [19]. The distinct feature of proposed NtruEncrypt-Light protocol is performing star multiplication operation in the ring R which is a cyclic convolution of two

polynomials of the same degree N. To achieve this, star multiplication library is implemented efficiently in code size.

AES is the preferred secret-key crypto algorithm in the library [7]. Block and encryption key lengths are fixed to 128-bits and CBC (Cipher Block Chaining) mode of operation is used for AES. One-way hashing operations have been executed using SHA-1 hash algorithm, which is implemented as standardized in FIPS 180-1.

Since we focus on the efficiency in memory usage and computation on sensor side, the software modules are grouped for each proposed protocol and the total size of code space is minimized. Modular multiplication based protocol and Modular multiplication-Light protocol libraries consist of Montgomery modular multiplication, SHA-1 and AES modules. Implemented code library is a total size of approximately 10 KB, including 5.5 KB for modular multiplication, 2.5 KB for SHA-1 and 2 KB for AES module. NtruEncrypt-Light key establishment protocol library is a total size of 5.5 KB, including 1 KB for star multiplication. MSP430 microcontroller authenticates the flash memory code before bootstrapping [27], so the code library can be kept in flash memory. We also need total size of 144 bytes of memory to store public key of command base, n_{CB}, and private response generation key e_S. For Light protocols, we need total size of only 32 bytes of memory for storage of e_S and u_S.

4.7.3 Computational and Communication Complexity

In this chapter, real-time executions of proposed protocols are analyzed individually. Selected operand size for modular multiplication is 1024 bits, which provides a security level of around 80 bits [17]. Sensor needs to make two modular squaring with 1024 bit modulus, five hashing operations, one decryption process, one MAC

generation, one random number generation and two exor operations for the modular multiplication based protocol.

The computational complexity of Montgomery product algorithm [14] with 1024-bit operand is approximately 2187 16-bit multiplications and 30719 16-bit additions. Implementation performs squaring with modulus only in 34.5 msec. The result is obtained with assistance of hardware multiplier for 16-bit multiplications. SHA-1 hashing algorithm takes approximately 1.0 msec. AES algorithm performs key scheduling in 1.1 msec and decrypts 256-bit ciphertext only in 2.5 msec in CBC mode. AES algorithm is also used for generation of message authentication code (MAC). Random number generation can be achieved by hashing hardware or system specific time-variant numbers. Using SHA-1 for the random number generation costs additional 1.0 msec. Exor operations take significantly small amount of time and we ignore the time taken for two exor operations.

Total processing time of modular multiplication based protocol on the MSP430 is only 82.3 msec and power consumption is approximately 28.6 mW. Total communication complexity of the scheme is 2656 bits.

In Modular multiplication-Light key establishment protocol sensor needs to make one modular squaring with 1024 bit modulus, four hashing operations, one decryption process, two MAC generations and one exor operation. Thus, sensor processing time for this scheme is 48.9 msec and power consumption is 17.0 mW while total communication complexity is 2784 bits. In the NtruEncrypt-Light protocol, $(N, q, p) = (263, 128, 3)$ is chosen as NtruEncrypt parameters to provide the same security level with 1024 bit modular multiplication [12]. To encrypt $m = (x \,||\, r_S \,||\, z_S)$, sensor needs to make 69169 small integer multiplications modulo q. Reducing the final polynomial modulo q is only discarding the least significant $\log_2 q = 7$ bits of

each coefficient. Thus, we can ignore the time elapsed for reduction modulo q. Implementation takes only 29.8 msec to complete the star multiplication. NtruEncrypt-Light scheme is slightly more efficient than other proposed schemes in terms of sensor processing time, but it looses out in terms of message size. Overall protocol takes approximately 45.2 msec on the MSP430 and consumes only 14.8 mW of battery power. However, total communication complexity of NtruEncrypt-Light key establishment protocol stays highest among others with 4418 bits.

4.7.4 Comparisons

In this chapter, the proposed protocols are compared with other public key based authenticated key establishment protocols in terms of computation and communication complexities.

Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) protocol [25], Aydos-Yanik-Koc protocol [2], and MSR-Hybrid protocol [14] are specifically selected because of their fast processing speeds and small message sizes in comparison with modular exponention based techniques. Note that these schemes are fully public key cryptography based and they use public key certificates for both sensor and station.

Comparison results are given in **Table 4-4**. All of given elliptic curve based protocols execute random point (RP) multiplications and/or fixed point (FP) multiplications. Choosing 160-bit key length ECC curve, one random point multiplication can be performed with 123832 16-bit multiplications and 1749704 16-bit additions [14]. Fixed point multiplication can be executed with much less work; 24143 16-bit multiplications and 341145 16-bit additions. As seen in **Table 4-4**, elliptic curve based protocols need much more processing time, computational capability and battery power than proposed modular multiplication based protocol. However, smaller

key sizes make elliptic curve based protocols more advantageous. We need minimum 1024-bit key length for Rabin's protocol and 1841-bit public key length for NtruEncrypt to meet the same security level with 160-bit ECC based protocols.

Although NtruEncrypt-Light scheme has less processing time on sensor, the scheme may not be viable option with its large messaging size. One solution to decrease the messaging traffic can be embedding the public key of station to the sensor. However, in dynamic environment sensor may need to store several public keys which cannot be memory efficient solution.

Table 4-4. Comparison of public key based key establishment protocols

	Small modular mult.	Large modular mult.	Elliptic curve point mult.	Sensor processing time (msec)	Message size (bits)
ECDHE	-	-	2 RP + 3 FP	1350	1796
MSR-Hybrid	-	1	3 FP	455	3682
Aydos-Yanik-Koc	-	-	1 RP + 2 FP	741	1730
Mod Mult based	-	2	-	82.3	2656
Mod Mult-Light	-	1	-	48.9	2784
NtruEncrypt-Light	69169	-	-	45.2	4418

4.8 Conclusion

Introducing security to the communication between sensor and data processing station requires more specialized approaches since sensors have constraints in battery power, communication and computational capabilities. Expensive public-key operations such

as modular exponentiation and elliptic curve point multiplication consume significant amount of sensor resources.

In this paper, three different authenticated key establishment protocols are proposed, combining public-key cryptography, secret-key cryptography and one-way hash functions. Modular multiplication based protocol uses only two modular multiplications at the sensor side and expensive public key decryption operation is executed at the station side.

Stations are implicitly certified to decrease the communication complexity. Proposed Modular multiplication-Light protocol improves the computational efficiency, and sensor needs to make only one modular multiplication. One of the modular multiplications which is used for verification of implicit certificate is replaced with inexpensive secret-key encryption. NtruEncrypt-Light protocol gives more efficiency in computation and battery consumption. It employs NtruEncrypt algorithm and replaces modular multiplication with one cyclic convolution product of two polynomials and polynomial addition. As a conclusion, proposed key establishment schemes provide strong security feature of public key cryptography with low battery power consumption and moderate level protocol communication.

4.9 References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102-114, August 2002.

[2] M. Aydos, T. Yanik, and C. Koc. An high-speed ECC-based wireless authentication protocol on an ARM Microprocessor. *Proceedings of ACSAC'2000*, New Orleans, LA, December 2000.

[3] S. Bakhtiari, R.Safavi-Naini, and J. Pieprzyk. Cryptographic hash functions: A survey. *Technical Report 95-09*, University of Wollongong, July 1995.

[4] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi. Secure pebblenets. Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001), 2001.

[5] H. Cam, S. Ozdemir, D. Muthuavinashiappan, and P. Nair. Energy-efficient security protocol for wireless sensor networks. *Proceedings of IEEE Conference on VTC*, 2003.

[6] Certicom Research, Standard for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 1.0, September 2000.

[7] J. Daemen and V. Rijmen. AES Proposal: Rijndael. AES algorithm submission, Sep 1999, at http://www.nist.gov/aes.

[8] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. *ACM CCS'03*, October 2003, Washington, DC, USA.

[9] L. Eschenauer and V.D. Gligor. A key-management scheme for distributed sensor networks. *Proceedings of the 9th ACM conference on Computer and communications security*, November 2002.

[10] G. Gaubatz, J. Kaps, and B. Sunar. Public key cryptography in sensor networks-revisited. *Proceedings of 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS'2004)*, 2004.

[11] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A ring-based public key cryptosystem. *Algorithmic Number Theory (ANTS III)*. Volume 1423 of LNCS., Pg.267-288, Berlin, Springer-Verlag, 1998.

[12] J. Hoffstein, J. Silverman, and W. Whyte. NTRU report 012, version 2. Estimated breaking times for NTRU lattices. Technical Report 12, NTRU Cryptosystems, Inc., Burlington, MA, USA, 2003.

[13] J. Hoffstein and J. H. Silverman. Optimizations for NTRU. *Proceedings of Public Key Cryptography and Computational Number Theory*, de Gruyter, Warsaw, 2000.

[14] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. *Proceedings of ACM WSNA'03*, September 2003, San Diego, CA, USA.

[15] ISO/IEC 9796. Information technology – Security techniques – Digital signature scheme giving message recovery, International Organization for Standardization, Geneva, Switzerland, 1991.

[16] G. Jolly, M. C. Kuscu, P. Kokate, and M. Younis. A low-energy key management protocol for wireless sensor networks. *Eighth IEEE International Symposium on Computers and Communications* (ISCC'03), 2003.

[17] A. K. Lenstra and E .R. Verheul. Selecting cryptographic key sizes. Journal of Cryptology. *The Journal of the International Association for Cryptologic Research 14*, Pg. 255-293, 2001.

[18] A. J. Menezes, P. C. Van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press Inc., 1997.

[19] P. L. Montgomery. Modular multiplication without trail division. *Mathematics of Computation*, Vol. 44, No. 170, pg. 519-521, 1985.

[20] B. C. Neuman and T. Tso. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32 (9). Pg.33-38, September, 1994.

[21] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler. SPINS: Security protocols for sensor networks. *Wireless Networks* 8, Pg. 521-534, The Netherlands, 2002.

[22] R. D. Pietro, L. V. Manchini, and A. Mei. Random key-assignment for secure wireless sensor networks. *Proceedings of the 1st ACM Workshop Security of Ad Hoc and Sensor Networks*, VA, USA, 2003.

[23] M.O. Rabin. Digitalized signatures and public key functions as intractable as factorization. Mit/lcs/tr-212, MIT, 1979.

[24] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, vol. 21, 120-126, 1978.

[25] SECG, Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, 2000. Available from http://www.secg. org/collateral/sec1.pdf.

[26] Texas Instruments, Inc. MSP430x4xx Family User's Guide, 2004. At http://www-s.ti.com/sc/psheets/slaz008/ slaz008.pdf.

[27] Texas Instruments, Inc. MSP430F42x Mixed Signal Controller, 2004. At http://www-s.ti.com/sc/ds/ msp430f427.pdf.

[28] D. S. Wang and A.H. Chan. Mutual authentication and key exchange for low power wireless communications. *Proceedings of IEEE MILCOM'2001*, 2001.

[29] H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory* 26, Pg.726-729, 1980.

4.10 Appendix A - NtruEncrypt Public Key Encryption Algorithm

An NTRU cryptosystem depends on three integer parameters (N, p, q) and four sets L_f, L_g, L_Φ, L_m of polynomials of degree N-1 with integer coefficients, where N is prime. The assumption is p and q are relatively prime numbers and q is considerably larger than p. NtruEncrypt is based on arithmetic in a polynomial ring $\mathbf{R = Z(x) / ((x^N-1), q)}$. Multiplication in the ring R is also called as "Star Multiplication" and denoted with the operator symbol '⊗'. Star multiplication is defined as cyclic convolution product of two vectors $f(x)$ and $g(x)$ in the following case:

$h(x)=f(x) \otimes g(x)$ with,

$$h_k = \sum_{i=0}^{k} f_i \cdot g_{k-i} + \sum_{i=k+1}^{N-1} f_i \cdot g_{N+k-i} = \sum_{i+j \equiv k (\mathrm{mod} N)} f_i \cdot g_j$$

Following the convolution product, each coefficient h_k is reduced with modulus p or q according to selection of encryption or decryption process.

Key Generation: Entity A chooses N, p, q, and generates its own private and public keys.

1. Choose random polynomials $F(x)$ and $g(x)$ from the ring R; both polynomials should have small coefficients.

2. Compute the private key $f(x) = 1 + pF(x)$, and inverse of $f(x)$, $f^{-1}(x)$ (mod q).

3. Compute the public key $h(x) \equiv g(x) \otimes f^{-1}(x)$ (mod q).

Encryption: Entity B encrypts the message m for A.

1. Encode the message into a polynomial $m(x)$ with binary or ternary coefficients.

2. Choose a random polynomial $\Phi(x)$ from the ring R.

3. Compute the ciphertext $c(x) \equiv p\Phi(x) \otimes h(x) + m(x) \pmod{q}$.

Decryption: Entity A decrypts the ciphertext message $c(x)$.

1. Compute the polynomial $m'(x) \equiv c(x) \otimes f(x) \pmod{q}$.

2. Compute the plaintext, $m(x) \equiv m'(x) \otimes f^{-1}(x) \pmod{p}$.

CHAPTER 5

HIGH-SPEED ECC BASED KERBEROS AUTHENTICATION PROTOCOL FOR WIRELESS APPLICATIONS

Özkan M. Erdem

School of Electrical Engineering and Computer Science

Oregon State University

Corvallis, OR 97331

Proceedings of IEEE Global Telecommunications Conference

Globecom'2003 Communication Security Symposium

San Francisco, CA, December 2003.

Abstract

We improved Kerberos authentication protocol for wireless communication using elliptic curve cryptographic operations. Proposed protocol offers the strength of public key cryptography and costs only 68.7 ms extra load to standard Kerberos without using pre-computation tables for 160-bit curve and scalable architecture, whereas with customized curve library it costs 57.3 ms extra load for the same key length. Using pre-computation tables reduces these timings to 55.8 ms and 51.6 ms respectively. Proposed protocol requires less bandwidth than other public key cryptography enabled Kerberos solutions. Results were obtained using 32-bit StrongARM1 processor runs 206 MHz. Due to its bandwidth efficiency and fast execution performance ECC supported Kerberos protocol can be an important option among other user authentication protocols for wireless networks.

5.1 Introduction

Today many network authentication protocols support public key cryptography to provide authentication between client and servers and to establish trusted connections. The documented IETF draft PKINIT proposed the usage of public key encryption and digital signature algorithms in Kerberos protocol to provide confidentiality, integrity and availability [10]. However, in wireless communication context although public key cryptography offers robust solutions to many security and authentication problems, the excessive amount of required computational resources and unacceptable performance characteristics remarkably limit the usage of public key cryptography. Public key cryptography can be integrated into Kerberos in a number of ways. Using Elliptic Curve Cryptography offers higher strength per key bit in comparison with other public key methods [1]. It has been claimed that computational power required to break 1024-bit of RSA is approximately equal to computational power to break 139-bit ECC [7]. Smaller bandwidth requirements and smaller certificate sizes due to shorter key sizes are other advantages of using ECC. However without efficient and optimized implementations, elliptic curve operations take excessive amount of time and require bigger code size than mobile platform can usually offer.

Chapter 5.2 provides brief background on Kerberos protocol and public key enabled version whereas Chapter 5.3 gives details of elliptic curve cryptography. In Chapter 5.4 we propose efficient protocol, which enables Kerberos to use elliptic curve operations. Chapter 5.5 gives details of our high speed and scalable implementation and specifications of used software and hardware. Following the development of test-bed, we analyze the performance characteristics for 32-bit

StrongARM1 processor based mobile devices and for Pentium-III servers in Chapter 5.6. Chapter 5.7 summarizes the conclusions of this study.

5.2 Kerberos and Its Derivations

Originated by MIT Project Athena, Kerberos is the distributed authentication service that provides data integrity and encryption services in specified networks. Standard protocol employs key distribution center (KDC), client and application server(s). As a default feature, protocol assumes both parties client and trusted server share the same secret key at the beginning of the protocol. Although basic purpose is verifying the identity of client for the application server, Kerberos also includes options for mutual authentication and establishment of additional secret key between parties. The detailed transaction flows of standard Kerberos v5 can be found in [10].

5.2.1 Public Key Cryptography Enabled Kerberos

Like in many security infrastructures, usage of public-key cryptography has been proposed in Kerberos protocol to provide confidentiality, integrity and availability. One of these proposals is documented in IETF draft, namely PKINIT [13]. PKINIT provides client authentication to KDC using public-key cryptography and digital signature algorithms. The main approach is to avoid using shared session key between client and KDC. Client signs his initial authentication service request with his private signing key and sends to KDC. KDC verifies the client and sends generated session key back to client after encrypting it with the public key of client. Public-private key pairs can be distributed with PKI supported environment to the clients. Adaptation of PKINIT to mobile users is also possible. However this adaptation has some important

drawbacks. Majority of mobile platforms have constraints in processing power in comparison with wired platforms. Communication restrictions such as bandwidth problem can be counted as other constraint of using public key cryptography in wireless devices. Using proxy servers however can decrease the processing load on mobile devices. Also in PKI enabled environment certificate chain can be cached by the proxy so that possible communication overhead with storing and forwarding certificate in proxy node could be mitigated. **Figure 5-1** shows the transaction flow of PKINIT protocol.

In [3] Harbitter and Menasce implemented the mobile adaptation of PKINIT. However this implementation has one difference regarding to generation of session key. Implementation employs clients to generate session key before sending authentication service request to KDC instead of waiting KDC to generate session key. This modification accommodates the operation of PKINIT in situations where the client only possesses a signing key, but can also be used with RSA that allows both signing and encryption with the same key and algorithm. The main reason for this swap is having mobile platform to do public key operation since private key operation requires more power resources for RSA type algorithms. Since session key is generated by mobile device, authors suppose to have one major potential risk in generation of strong secure session key.

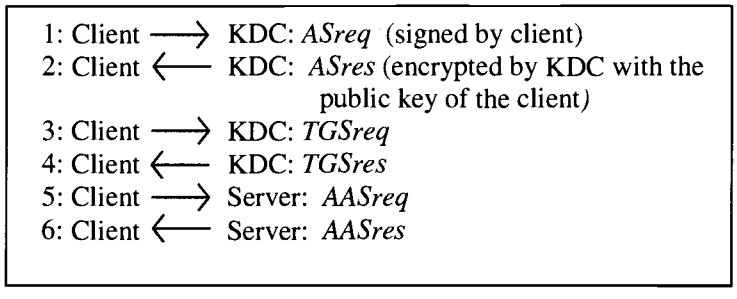

Figure 5-1. Protocol flow for PKINIT

Table 5-1. Notation

Symbol	Meaning	Symbol	Meaning
dS_C QS_C	Client's private and public key pair for sign/verify	$Cert$	Certificate
dE_C QE_C	Client's private and public key pair for encryption/decryption	d_{KDC} Q_{KDC}	Private and public key pair for KDC
E(k, m)	Secret-key encryption of m under the key k	I_C I_S	Identity information
k_C, k	Selected random values	N_C, N_{KDC}	Nonce values
$msb_{128}(m)$	Most significant 128 bits of m	$Sign_x(m)$	ECDSA signature of m under the key x
t_C	Expiration time for certificate	L	Lifetime value
(r_C, s_C)	ECDSA signature pair	$Q_{C\text{-}KDC}$	Key agreement key
H(m)	Hashing output of m	$K_{KDC\text{-}C}$	Encryption session key
Sk	Session key	$Auth_C$	Authenticator
x	ECC point multiplication operation	TK_S	Ticket for server S

5.3 Elliptic Curve Cryptography

Cryptographic operations on elliptic curves have two major advantages in comparison to other public-key cryptographic methods; high speed in operations with shorter key sizes and so less memory and bandwidth requirements in mobile platforms. The IEEE standard [5] proposes the use of the GF (p) and GF (2^k) fields for modular arithmetic on curves. Here GF (p) shows Galois field of prime modulus p and GF (2^k) shows Galois field of composite number, which is power of 2. The security of curve operation requires at least 100 bits for key size, preferably 160 bits and more for stronger keys.

Elliptic Curve Digital Signature Algorithm (ECDSA) is the analogue of Digital Signature Algorithm utilizes the arithmetic of points, which are elements of the set of solutions of an elliptic curve equation defined over a finite field, [5]. After definition of elliptic curve E over GF (p) with large group of order n and selection of a point P

of this large order, each party performs individual public-private key pair generation steps. The selected point P is the public information to all users. The primitives generated during finite field and base point selection process are the primitives of signature and verification operations as well.

5.4 ECC Enabled Kerberos

The main goal of using public key cryptography in Kerberos is to provide non-repudiation and mutual authentication services for client-KDC communications. PKINIT utilizes Diffie-Hellman key exchange in combination with RSA keys as the primary to provide non-repudiation and mutual authentication. Note that public and private key operations performed in factorization or in discrete logarithm based algorithms consume a significant amount of processing resources and time, naturally this causes negative effects on performance and response time of communicating parties. On the contrary, using elliptic curve operations not only reduces the amount of resources required by public-private key operations but also uses communication channel more effectively with the shorter key sizes [1,8]. Proposed authentication protocol defines elliptic curve E over $GF(p)$ with large group of order n and utilizes ECDSA algorithm and point arithmetic on E. However our design targets to have some major goals;

- Preserve the main semantics of Kerberos.
- Provide mutual authentication between client and KDC.
- Provide non-repudiation of client to KDC.
- Minimize the number of operations to be performed on mobile client.
- Keep ability to use existing or developing public key management infrastructures.

Our proposal assumes using certificates to identify and authenticate parties, although this does not oblige to have existing public key infrastructure. Since KDC is the trusted party, storage of public key and certificates by KDC is sufficient solution to manage keys.

5.4.1 Client and KDC Initializations

Initialization between KDC and client is one-time off-line process to be performed before using authentication protocol. First, as a trusted server KDC produces his private and public key pair for key agreement purposes and publishes to all clients as in follows:

One-time Setup for KDC:

1. Select an elliptic curve E over $GF(p)$ with a group of order n, choose a point P of this order.
2. Select a random integer $d_{KDC} \in [2, n\text{-}2]$.
3. Compute $Q_{KDC} = d_{KDC} \times P$.
4. The public and private keys of KDC are (E, P, n, Q_{KDC}) and d_{KDC} respectively. Public key of KDC will be published to all clients.

The next step is two-way certificate initialization for each client. Client chooses private signature generation key dS_C, private key agreement key dE_C, and initially set identity information I_C. After computing signature verification key QS_C and public key agreement key QE_C client generates certificate request message and sends to KDC. Key setup and certificate generation are illustrated briefly in **Figure 5-2**.

KDC stores client's public key agreement key, signs the certificate including signature verification key, identity information and the expire date of certificate t_C and sends certificate back to client. Here r_C is the x coordinate of the elliptic curve point R_C, and

H is the hash function. The application server repeats the same process to acquire its certificate like shown in **Figure 5-2**.

1: Client: Select $dS_C \in [2, n\text{-}2]$. Compute $QS_C = dS_C \times P$.
Select $dE_C \in [2, n\text{-}2]$. Compute $QE_C = dE_C \times P$.
2: Client $\longrightarrow$ **KDC:** ***CERTreq***: $\{I_C, QS_C, QE_C\}$.
3: KDC: Select $k_C \in [2, n\text{-}2]$.
Compute $R_C = k_C \times P$.
Compute $m = \{I_C \mid QS_C.x \mid t_C\}$.
Use ECDSA to sign m, compute

$$r_C = R_C.x,$$
$$s_C = k_C^{-1}.(\mathrm{H}(m)+d_{KDC}.r_C)$$
$$Cert_C = \{m, t_C, (r_C, s_C)\}$$

4: Client $\longleftarrow$ **KDC:** ***CERTres***: $Cert_C$
5: Client: Store the $Cert_C$ after verification of its signature by using ECDSA algorithm with public key of KDC.
If not verified go to 1.
Compute and store public key agreement key.

$$Q_{C\text{-}KDC} = dE_C \times Q_{KDC}$$

Store dS_C as a signature generation key and dE_C as a private key agreement key.
6: KDC: Store identity of client I_C and QE_C as the public key agreement key of client.
Store the $Cert_C$.

Figure 5-2. Proposed protocol flow for key setup and certificate generation

5.4.2 Design of Proposed Protocol

One of our goals is preserving Kerberos semantics so that proposed protocol would be used with existing standard Kerberos KDC and application server implementations. Protocol starts with the request for authentication and ticket service (*ASreq-TGSreq*) signed by client using private key *dSc* with ECDSA algorithm. Standard Kerberos v5 message contains a PA-PK-AS-SIGN pre-authentication field, which includes sub fields the "*userCert*" and "*encSignedRandomKeyPack*". These sub fields can be used

for signature and certificate delivery to KDC. After verification of the client's signature KDC generates encryption key agreement key $Q_{C\text{-}KDC}$ shared between client and KDC. For this, KDC performs point multiplication operation using d_{KDC} and QE_C. Result gives $Q_{C\text{-}KDC}$ shared by both parties whereas client already did the same operation during initialization.

1: Client: Compute $m = \{I_C \mid I_S \mid N_C\}$.
2: Client ⟶ **KDC:** ***ASreq-TGSreq*:** $\{m \mid Sign_{dSc}(\mathrm{H}(m))\}$.

3: KDC: One-time setup: $TK_S = \mathrm{E}(K_{KDC\text{-}S}, k \mid I_C \mid L)$
4: KDC: Verify the signature of *ASreq-TGSreq* using $Cert_C$.
If verified;
{ $Q_{C\text{-}KDC} = d_{KDC}$ x QE_C
$K_{KDC\text{-}C} = msb_{128}(\mathrm{H}(Q_{C\text{-}KDC} \mid N_C \mid N_{KDC}))$
$\mathrm{E}(K_{KDC\text{-}C}, k \mid N_C \mid N_{KDC} \mid L \mid I_S)$
}
5: Client ⟵ **KDC:** ***ASres-TGSres*:**
$\{TK_S \mid \mathrm{E}(K_{KDC\text{-}C}, k \mid N_C \mid N_{KDC} \mid L \mid I_S)\}$
6: Client: Check N_C, Generate
$K_{KDC\text{-}C} = msb_{128}(\mathrm{H}(Q_{C\text{-}KDC} \mid N_C \mid N_{KDC}))$
$Auth_C = \mathrm{E}(k, I_C \mid t_C \mid Sk_C)$
7: Client ⟶ **Server:** ***AASreq*:** $TK_S \mid Auth_C$
8: Server: Decrypt TK_S, retrieve k
Decrypt $Auth_C$.
Check I_C and L.
Authentication successful.
9: Client ⟵ **Server:** ***AASres:*** $\mathrm{E}(k, t_C \mid$ (optional) $Sk_S)$.
(optional) Key agreement with Sk_C and Sk_S.

Figure 5-3. Proposed ECC based Kerberos protocol

One-time encryption session key is derived from the digest of the concatenation of key agreement key ($Q_{C\text{-}KDC}$), nonce generated by client (N_C) and nonce generated by KDC (N_{KDC}). Since the recent symmetric encryption algorithm standards assume using

minimum 128 bits key size we suggest using the most significant 128 or more bits of digest as symmetric encryption session key $K_{KDC\text{-}C}$. Authentication and ticket granting service response message (*ASres-TGSres*) has additional N_{KDC} field. Finally client uses this nonce value to compute session encryption key. The rest of protocol will be very same as specified in standard Kerberos v5.

The additional computational cost of protocol on the mobile user side is just one signature generation and two hashing operations where signature generation means a point multiplication on the curve, one hashing and one modular inverse calculation. Note that client should store the seed key ($Q_{C\text{-}KDC}$) safely inside of device and use it in every new transaction. In next chapter we briefly talk about the structure of implementation to test our proposed protocol, the scalability and the timing figures. **Figure 5-3** gives details on the protocol.

5.5 Structural Implementation

Implemented configuration consists of simple communication entities in Kerberos protocol; Key distribution center, application server and mobile client. Two different Ethernet based LANs have been used to simplify the performance evaluation. The configuration also includes three client workstations locating on each individual LAN to load the KDC in Ethernet LAN1 and application server in Ethernet LAN2 by performing large number of authentication transactions. Since usually the link between KDC and Application server runs on wide area network we simulated WAN connection between two LANs. However high scoped systems employ most likely more powerful computing resources and processors in KDC and application server than the one we used in the configuration. Mobile client uses Windows CE operating system as the majority of the handheld computer and PDA devices support. The KDC

and application server run Windows 2000 server operating system whereas client workstations use Windows 2000 professional OS. In our configuration we don't see the necessity of using proxy server. The simplified transactions of mobile client and the efficiency of point multiplication and signature generation processes in ECC can be counted as the reasons for not to employ proxy server. Structure of the performance evaluation and test configuration briefly figured in **Figure 5-4**.

5.5.1 Software and Hardware Specifications

Elliptic point addition, doubling, point multiplication and ECDSA algorithm have been implemented as standardized in the IEEE P1363 and ANSI X9.62. We preferred to use DES, 3DES and AES as secret-key crypto algorithms. We also have used SHA-1 message digest algorithm to hash the message blocks. All these algorithms have been implemented as specified in FIPS [46-3, 81], FIPS [46-3], FIPS [197] and FIPS [180-1] standards, respectively.

Implemented modular and point arithmetic module performs modular operations, such as addition, subtraction, multiplication, inversion operations for modulus p, as well as point addition, doubling and point multiplication on the elliptic curve. This small sized module is implemented in only 28 kilobytes of code size for random curve selections and in 44 kilobytes of code size for customized NIST curve (256 bit key length). It optionally uses a small amount of extra memory and provides speed-up in operations. Module supports all standard ECC fields, random base points, random curve parameters and works for different elliptic curves with various key lengths from 160 bits up to 256 bits.

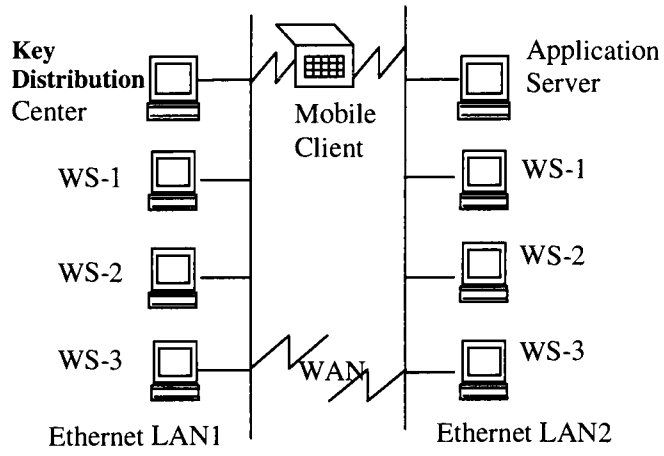

Figure 5-4. Performance measurement and test configuration

Implemented ECDSA module performs elliptic curve parameter and key generation as well as it generates signature of hashed messages and verifies the signature obtained. Today there are plenty of pocket PC and handheld PC currently employ Intel SA-1110 processor. In our configuration we used a highly integrated 32-bit Intel StrongARM 206 MHz processor that incorporates Intel design and process technology along with the power efficiency of the ARM architecture, [6, 14]. As KDC and application server we used two servers running 850 MHz Pentium III processors. All Pentium processors used are little endian.

5.6 Performance Analysis

In this chapter we analyze the service time measurements for the proposed protocol. As an initial step we obtained the performance results for ECC operations including elliptic curve point multiplication, ECDSA signature generation and signature verification. Firstly selected three curves are random NIST curves with the key length

160, 192 and 256 bits. These curves use our general-purpose implemented code library that provides simultaneous support at runtime for each of these curves. Secondly selected three curves use specifically optimized different code libraries. Field multiplication, modulo reduction and other curve-specific parts have been optimized for these customized curves whereas the core of ECC library has been kept as is. Since the code size and memory heap usage are major criteria in mobile devices, we briefly give the code sizes of the library and runtime memory usage in **Table 5-2** for each curves and algorithms. For each curve in the table, code sizes and memory usage are provided without using pre-computation and with using pre-computation options as well. Pre-computation option generates "pre-computation tables" and stores them in the memory during curve initialization phase to speed-up the point multiplication and signature generation operation for the specified curve. Note that using pre-computation tables causes to increase in code size about 4 KB and four to seven times bigger memory heap usage in runtime. Protocol transaction steps have been grouped to two main segments to analyze the service time clearly:

Pre-Auth: Measured time for initialization of the mobile device and transmission of *ASreq-TGSreq* to KDC prior to authentication.

Auth: Measured time for KDC computations and transmission of *ASres-TGSres* back to the mobile device, additionally the measured time for mobile device processing time of *ASres-TGSres* and completing the authentication with application server. The performance of each component in our simulation might show the variance from real time mobile computing environment. As a transmission speed on LAN we figured 10Mbps that is average speed of Ethernet LAN. We used 9600 bits per second wireless network rate to represent 2G network performances [12]. **Table 5-3** summarizes the additional service time for standard Kerberos to complete the protocol.

Table 5-2. Memory heap size and code size of crypto library implementation

Curves vs. Sizes (KB)	ECC 160 bit wop/wp	ECC 192 bit wop/wp	ECC 256 bit wop/wp
Code Size	28.0/32.0	28.0/32.0	28.0/32.0
Memory Usage	2.2/8.7	2.2/11.5	2.8/20.0
Curves vs. Sizes (KB)	NIST 160 bit wop/wp	NIST 192 bit wop/wp	NIST 256 bit wop/wp
Code Size	32.0/40.0	36.0/40.0	44.0/48.0
Memory Usage	2.5/8.5	2.8/11.5	3.6/20.0
	DES	TDES	AES
Code Size	14.0	14.0	27.0

Results have been obtained separately with using pre-computation tables (wp) and without using pre-computation tables (wop) on the code library. As seen in **Table 5-3**, using pre-computation improves the performance of ECC enabled Kerberos service timings. However extra load in memory of mobile device is a drawback of using pre-computation.

Several speed optimizations on the library can be done easily with machine level coding. In our implementation we obtained approximately twenty percent speed-up in point multiplication and so in signature generation/verification by coding in ARM assembly machine language. These optimizations mainly performed in low-level field multiplication that is taking majority of processing time in point multiplication. We expect our server capacity to multiply with a factor of ten over the PCs that we used in our test-bed since today most of high-end servers employ much powerful CPUs. Another improvement on protocol service time can be increasing the network throughput to 3G speeds.

Table 5-3. Extra service time required for protocol

Protocol Extra Load (ms)	ECC 160 bit wop/wp	ECC 192 bit wop/wp	ECC 256 bit wop/wp
Pre-Auth.	53.0/41.6	71.7/51.2	124.3/79.8
Auth.	15.7/14.2	17.2/14.9	21.3/16.3
Total	68.7/55.8	88.9/66.1	145.6/96.1
	NIST 160 bit wop/wp	NIST 192 bit wop/wp	NIST 256 bit wop/wp
Pre-Auth.	42.9/37.6	55.2/46.5	88.8/66.9
Auth.	14.4/14.0	14.8/14.2	17.7/15.1
Total	57.3/51.6	70.0/60.7	106.5/82.0

5.7 Conclusions

In this paper we briefly explained how we provide ECC support to Kerberos authentication protocol.

Preserve the main semantics of Kerberos: In our proposal we have minor additions to standard Kerberos protocol. Mobile device can use pre-authentication fields for signature and certificate delivery to KDC.

Provide mutual authentication between client and KDC & Provide non-repudiation of client to KDC: Signature of *ASreq-TGSreq* proves the origin and integrity of the message to KDC. Authentication of the server is provided with the previously agreed key between KDC and client to derive the session key.

Minimize the number of operations to be performed on mobile client: PKINIT requires one RSA signature generation and one RSA decryption operation in client side during authentication process. We reduced the operations for preparation of *AS-TGS* request messages in client side. Only with one ECDSA signature generation mobile device completes the client side of mutual authentication process.

Keep ability to use existing or developing public key management infrastructures: Proposed protocol fully supports to use of certificates. Extension fields defined in Kerberos v5 can be used to transmit the certificates while the current IETF draft for PKINIT allows the KDC to store client private keys.

The measurements show that ECC enabled Kerberos offers very reasonable timings for mobile environment. Proposed protocol costs only 68.7 ms extra load to standard Kerberos without using pre-computation tables for 160-bit curve and scalable architecture, whereas with customized curve library it costs 57.3 ms extra load for the same key length. Using pre-computation tables reduces these values to 55.8 ms and 51.6 ms respectively. Wireless network speed has been presumed as 9600 bps as it is default speed in G2 networks. Finally, due to its bandwidth efficiency and fast execution performance ECC-supported Kerberos protocol can be an important option among other authentication protocols for wireless networks.

5.8 Acknowledgement

This research has been supported by ICEsoft Technologies.

5.9 References

[1] M. Aydos, T. Yanik, and C.K. Koc. High speed implementation of an ECC-based wireless authentication protocol on an ARM microprocessor. *IEE Proceedings: Communications,* 148(5): 273-279, October 2001.

[2] A. Fox, and S.D. Gribble. Security on the move: Indirect authentication using Kerberos. *MOBICOM'96.* Rye, NY, 1996.

[3] A. Harbitter, and D.A. Menasce. The performance of public key-enabled Kerberos authentication in mobile computing applications. *ACM, CCS'01*, Philadelphia, PA, November 2001, pp.78-85.

[4] M. Hur, et al. Public key cryptography for cross-realm authentication in Kerberos. http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pkcross-06.txt, 2000.

[5] IEEE P1363: Standard Specifications for Public-key Cryptography, Draft version 13, November 1999.

[6] Intel Corp., Intel StrongARM SA-1110 Microprocessor: Brief Datasheet, Santa Clara, California, 2001.

[7] A.K. Lenstra, and E.R. Verheul. Selecting cryptographic key sizes. *The 3rd Workshop on Elliptic Curve Cryptography (ECC'99)*, November 1999, Waterloo, Canada, pp. 1-3.

[8] A.J. Menezes. Elliptic Curve Cryptosystems. Kluwer Academic Publishers, Boston, MA, 1993.

[9] V. Miller. Uses of elliptic curves modulo large primes. *Advances in Cryptology, CRYPTO'85*, Lecture Notes in Computer Science, Springer-Verlag, 1986, pp.417-426.

[10] MIT, Kerberos: The Network Authentication Protocol, 1998, http://web.mit.edu/kerberos/www/.

[11] P. L. Montgomery. Modular multiplication without trivial division. *Mathematics of Computations*, April 1985, 44, (170), pp. 519-521.

[12] Personal Communications Industry Association. Market Demand Forecast for Terrestrial Third Generation (IMT-2000) Service for the PCIA, 1998.

[13] B. Tung, et al. Public Key Cryptography for Initial Authentication in Kerberos. http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt, 2001.

[14] R. Witek and J. Montanaro. StrongARM: A high performance ARM processor. *Proceedings of the COMPCON Spring'96-41st IEEE International Computer Conference*, Digital Equipment Corp., USA, 1996.

CHAPTER 6

GENERAL CONCLUSION

In this thesis three different open problems in mobile ad hoc network key management area are studied and novel solutions are proposed. Distributed trust model for mobile ad hoc networks does not employ an online key distribution server and devices do not have trust relations prior to forming the group. Self-organizing devices have to authenticate each other and establish the encryption link keys. A practical and efficient group key management framework is presented for resource constrained mobile ad hoc network to solve this problem. The proposed Hierarchical Binary Tree based model does not require any central key management authority or group manager, and uses one-way hash function and simple xor operations to generate the group meeting key. Every device has the same level of trust in the group and establishes the trust with other devices which locate in the same Key Sharing Subgroup. To complete the authentication, each user contacts to $d=\log_2 n$ member in join process, where n shows the number of members. Then, sibling member of the joining or leaving member in the tree generates the seed key which will be used in refreshing group key. Group key is the key of the root node which is result of consecutive hash functions applied to the seed key in upward direction. Distribution of group keys to other members is performed after encrypting the keys with previously negotiated blinded keys and index values. Every member holds $(2d + 1)$ keys, d indexes and d public keys for the group key management. Chapter 2 and Chapter 3 present two different methods to establish Key Sharing Subgroups. The group authentication and group key establishment protocols proposed in Chapter 2 employs any type of public key algorithms to complete the device authentication process while Chapter 3 presents a

customized ECC and Rabin's scheme based group authentication and key establishment protocol. Thus, more efficiency in the group key management is achieved during modifications in memberships. According to the comparisons, proposed customized protocols achieve better results than other recently proposed ECC based authenticated key establishment protocols in terms of processing time. Since there are $d=\log_2 n$ number of key establishment process for each joining device, minimizing total processing time significantly improves the efficiency of proposed group key management system.

It has also been proven that proposed group key management system satisfies the security requirements of common group philosophy, such as backward secrecy, forward secrecy, group confidentiality and key freshness. Consequently, proposed Hierarchical Binary Tree based system offers practical distributed key management for resource constrained systems which has lack of key distribution authority and achieves comparable performance with the models which use secret key cryptography and employ centralized authorities.

Chapter 4 addresses the problem of introducing public key cryptography to the communication between sensor and data processing station. Using expensive public-key operations such as modular exponentiation and elliptic curve point multiplications should be avoided since sensors have strict constraints in computational capabilities, energy and memory. Three different authenticated hybrid key establishment protocols are presented. Proposed modular multiplication based protocol uses only two modular multiplications at the sensor side and expensive public key decryption operation is executed at the station side. On the other hand, proposed Modular multiplication-Light protocol improves the computational efficiency, and sensor needs to make only one modular multiplication. Next proposed protocol NtruEncrypt-Light gives more

efficiency in computation and battery consumption. It employs NtruEncrypt algorithm and replaces modular multiplication with one cyclic convolution product of two polynomials and polynomial addition. Chapter 4 also gives the performance comparisons between proposed hybrid protocols and fully public key based protocols. In summary, hybrid authenticated key establishment protocols provide strong security feature of public key cryptography with significantly reduced processing time, low battery power consumption and comparable message size.

Chapter 5 briefly explains how to provide ECC support to Kerberos authentication protocol to solve the mutual authentication and to prevent password guessing attacks. Proposed certification and authentication protocols have only minor additions to standard Kerberos protocol and so standard semantics of Kerberos is preserved. Non-repudiation of client and authentication between client and Key Distribution Center are provided and freshness of session keys is guaranteed. While PKINIT [52] requires one RSA signature generation and one RSA decryption operation in client side during authentication, in proposed authentication protocol mobile device completes the client side of the process only with one ECDSA signature generation. The measurements show that ECC enabled Kerberos offers very reasonable message bandwidth and fast execution performance for mobile environment. As a result, ECC supported Kerberos protocol can be an important option among other authentication protocols for mobile ad hoc networks which prefer to use centralized trust model.

BIBLIOGRAPHY

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102-114, August 2002.

[2] M. Aydos, T. Yanik, and C. Koc. An high-speed ECC-based wireless authentication protocol on an ARM Microprocessor. *Proceedings of ACSAC'2000*, New Orleans, LA, December 2000.

[3] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Cryptographic hash functions: A survey. Technical Report 95-09, University of Wollongong, July 1995.

[4] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi. Secure pebblenets. Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001), 2001.

[5] H. Cam, S. Ozdemir, D. Muthuavinashiappan, and P. Nair. Energy-efficient security protocol for wireless sensor networks. *Proceedings of IEEE Conference on VTC*, 2003.

[6] S. Capkun, L. Buttyan, and J. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52-64, January-March, 2003.

[7] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner. Efficient security for large and dynamic multicast groups. *Proceedings of the Seventh Workshop on Enabling Technologies (WET ICE'98)*, IEEE Computer Society Press, 1998.

[8] J. Daemen and V. Rijmen. AES Proposal: Rijndael. AES algorithm submission, Sep 1999. http://www.nist.gov/aes.

[9] Certicom Research, Standard for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 1.0, September 2000.

[10] R. Di Pietro, L. V. Mancini, and S. Jajodia. Efficient and secure keys management for wireless mobile communications. *Proceedings of POMC'02, ACM*, Toulouse, France, October 2002.

[11] R. Di Pietro, L. V. Manchini, and A. Mei. Random key-assignment for secure wireless sensor networks. *Proceedings of the 1st ACM Workshop Security of Ad Hoc and Sensor Networks*, VA, USA, 2003.

[12] L. R. Dondeti, S. Mukherjee, and A. Samal. A distributed group key management scheme for secure many-to-many communication. *Technical Report PINTL-TR-207-99*, Department of Computer Science, University of Maryland, 1999.

[13] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. *ACM CCS'03*, October 2003, Washington, DC, USA.

[14] O. M. Erdem. High-speed ECC based Kerberos authentication protocol for wireless application. *Proceedings of IEEE GLOBECOM'2003 Communication Security Symposium*, San Francisco, CA, December 2003.

[15] O. M. Erdem. EDKM: Efficient distributed key management for mobile ad hoc networks. *Proceedings of IEEE International Symposium on Computers and Communications, ISCC'04*, Alexandria, Egypt, July 2004.

[16] O. M. Erdem. Efficient self-organized key management for mobile ad hoc networks. *Proceedings of IEEE GLOBECOM'2004 Network and Security Management Symposium*, Dallas, TX, December 2004.

[17] O. M. Erdem. Lightweight key establishment protocols for self-organizing sensor networks. To be submitted to *IEEE 2nd Annual Communication Society Conference on Sensor and Ad Hoc Communications and Networks, SECON'05*, Santa Clara, CA, September 2005.

[18] L. Eschenauer and V.D. Gligor. A key-management scheme for distributed sensor networks. *Proceedings of the 9th ACM conference on Computer and communications security*, November 2002.

[19] A. Fox and S.D. Gribble. Security on the move: Indirect authentication using Kerberos. *MOBICOM'96*. Rye, NY, 1996.

[20] G. Gaubatz, J. Kaps, and B. Sunar. Public key cryptography in sensor networks-revisited. *Proceedings of 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS'2004)*, 2004.

[21] S. Gokhale and P. Dasgupta. Distributed authentication for peer-to-peer networks. *International Symposium on Applications and the Internet (SAINT) 2003*, Pg. 347-353, Orlando, FL.

[22] A. Harbitter and D.A. Menasce.The performance of public key-enabled Kerberos authentication in mobile computing applications. *ACM, CCS'01*, Philadelphia, PA, November 2001, Pg. 78-85.

[23] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A ring-based public key cryptosystem. *Algorithmic Number Theory (ANTS III)*. Volume 1423 of LNCS., Pg. 267-288, Berlin, Springer-Verlag, 1998.

[24] J. Hoffstein, J. Silverman, and W. Whyte. NTRU report 012, version 2. Estimated breaking times for NTRU lattices. *Technical Report 12*. NTRU Cryptosystems, Inc., Burlington, MA, USA, 2003.

[25] J. Hoffstein and J. H. Silverman. Optimizations for NTRU. *Proceedings of Public Key Cryptography and Computational Number Theory*, de Gruyter, Warsaw, 2000.

[26] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. *Proceedings of ACM WSNA'03*, September 2003, San Diego, CA, USA.

[27] M. Hur, et al. Public key cryptography for cross-realm authentication in Kerberos. http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pkcross-06.txt, 2000.

[28] IEEE P1363: Standard Specifications for Public-key Cryptography, Draft version 13, November 1999.

[29] ISO/IEC 9796. Information technology – Security techniques – Digital signature scheme giving message recovery, International Organization for Standardization, Geneva, Switzerland, 1991.

[30] Intel Corp., Intel StrongARM SA-1110 Microprocessor: Brief Datasheet, Santa Clara, California, 2001.

[31] G. Jolly, M. C. Kuscu, P. Kokate, and M. Younis. A low-energy key management protocol for wireless sensor networks. *Eighth IEEE International Symposium on Computers and Communications* (ISCC'03), 2003.

[32] A. Khalili, J. Katz, and W. Arbaugh. Toward secure key distribution in truly ad-hoc networks. *International Symposium on Applications and the Internet (SAINT) 2003*, Pg. 342-346, Orlando, FL, 2003.

[33] A. K. Lenstra and E .R. Verheul. Selecting cryptographic key sizes. Journal of Cryptology. *The Journal of the International Association for Cryptologic Research 14*, Pg. 255-293, 2001.

[34] A.K. Lenstra and E.R. Verheul. Selecting cryptographic key sizes. *The 3rd Workshop on Elliptic Curve Cryptography (ECC'99)*, November 1999, Waterloo, Canada, Pg. 1-3.

[35] D. A. McGrew and A.T. Sherman. Key establishment in large dynamic groups using one-way function trees. *Technical Report No.0755*, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[36] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of applied cryptography. *CRC Press series on discrete mathematics and its applications*. CRC Press, 1997.

[37] A.J. Menezes. Elliptic Curve Cryptosystems. Kluwer Academic Publishers, Boston, MA, 1993.

[38] V. Miller. Uses of elliptic curves modulo large primes. *Advances in Cryptology*, CRYPTO'85, Lecture Notes in Computer Science, Springer-Verlag, 1986, Pg. 417-426.

[39] MIT, Kerberos: The Network Authentication Protocol, 1998. http://web.mit.edu/kerberos/www/.

[40] P. L. Montgomery. Modular multiplication without trail division. *Mathematics of Computation*, Vol. 44, No. 170, Pg. 519-521, 1985.

[41] B. C. Neuman and T. Tso. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32 (9). Pg. 33-38, September, 1994.

[42] A. Perrig, D. Song, and J. D. Tygar. ELK, a new protocol for efficient large-group key distribution. *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2001.

[43] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler. SPINS: Security protocols for sensor networks. *Wireless Networks* 8, Pg. 521-534, The Netherlands, 2002.

[44] Personal Communications Industry Association, Market Demand Forecast for Terrestrial Third Generation (IMT-2000) Service for the PCIA, 1998.

[45] M. O. Rabin. Digitalized signatures and public key functions as intractable as factorization. Mit/lcs/tr-212, MIT, 1979.

[46] S. Rafaeli, L. Mathy, and D. Hutchison. EHBT: An efficient protocol for group key management. *Proceedings of the Third International COST264 Workshop (NGC 2001)*, Springer-Verlag, LNCS 2233, Pg. 159-171, London, UK, November 2001.

[47] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol. 21, Pg. 120-126, 1978.

[48] SECG, Elliptic Curve Cryptography, Standards for Efficient Cryptography Group, 2000. Available from http://www.secg. org/collateral/sec1.pdf.

[49] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A new approach to group key agreement. *Technical Report RZ 2984*, IBM Research, December 1997.

[50] Texas Instruments, Inc. MSP430x4xx Family User's Guide, 2004. At http://www-s.ti.com/sc/psheets/slaz008/slaz008.pdf.

[51] Texas Instruments, Inc. MSP430F42x Mixed Signal Controller, 2004. At http://www-s.ti.com/sc/ds/msp430f427.pdf.

[52] B. Tung, et al. Public Key Cryptography for Initial Authentication in Kerberos, 2001. http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt.

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

[54] D. S. Wang and A.H. Chan. Mutual authentication and key exchange for low power wireless communications. *Proceedings of IEEE MILCOM'2001*, 2001.

[55] H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, 26 (1980), Pg. 726-729.

[56] R. Witek and J. Montanaro. StrongARM: A high performance ARM processor. *Proceedings of the COMPCON Spring'96-41st IEEE International Computer Conference*, Digital Equipment Corp., USA, 1996.

[57] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6), November-December 1999.