Implementation of Group Member Authentication Protocol in Mobile Ad-hoc Networks

Hitoshi Asaeda*, Musfiq Rahman†, Mohammad Hossein Manshaei‡, Yasuko Fukuzawa§
*Keio University, Graduate School of Media and Governance, 5322, Endo, Fujisawa-shi, Kanagawa-ken, Japan
†Asian Institute of Technology, P.O. Box 4, Klong Luang, Pathumthani, Thailand
‡INRIA, Planete Research Team, 2004, Route des Lucioles, BP 93, Sophia Antipolis, France
§Hitachi Ltd., Systems Development Laboratory, 1099, Ouzenji, Asou-ku, Kawasaki-shi, Kanagawa-ken, Japan

***Abstract*— In a mobile ad-hoc network (MANET) architecture, there is no pre-existing fixed network infrastructure, and a mobile node in this network sends data packets to a destination node directly or through its neighbor nodes. This situation is of potential security concern since the neighbor nodes cannot be always trusted. In this paper, we design a group member authentication protocol used in a MANET. It aims to allow a set of nodes to legitimately participate in group communication and then distribute a secret group key to the approved nodes to establish secure communication with group members. Our protocol provides knowledge-based group member authentication, which recognizes a list of secret group keys held in a mobile node as the node's group membership. It employs Zero Knowledge Proof and threshold cryptography. We then introduce our actual implementation and evaluate the behavior to ensure its successful deployment.**

I. Introduction

Communication via a mobile ad-hoc network (MANET) is fundamental to ubiquitous wireless networking. It enables easy and instantaneous communication between two or more nodes without the aid of infrastructure or centralized administration. The network topology is dynamically formed and the connectivity among the nodes may vary with time due to departures, arrivals and movements of nodes.

In an ad-hoc manner, each mobile node discovers its neighbor nodes and then establishes communication paths with them. In this environment, a mobile node does not distinguish its neighbor nodes in general, and hence its data packets may go through arbitrary nodes to reach the destination when the destination nodes are not within the same radio range. This situation is of potential security concern to a node that wishes to communicate with its intended neighbor nodes since some other malicious user within the range or routing path can eavesdrop on its data packets or send bogus traffic that causes denial-of-service attacks.

While MANET communication has become an attractive option for wide-scale commercial use, its lack of innate security impedes actual deployment and further growth. A related concern is that, unlike communication over the Internet, traditional security mechanisms such as authentication protocols and data encryption techniques cannot be easily adapted in a MANET because any certification authority or key distribution infrastructure does not exist in the network. This situation is in sharp contrast to other mobile networking platforms such as Mobile IP or cellular telephony where nodes are always under some administrative direction – in spite of their mobility.

Based on the characteristics of a transiently associated network, separating neighbor nodes into trusted and non-trusted groups is crucial. In general, a security incident in a MANET does not belong to any individual, but rather binds to a group like a company or a project. In other words, the provision against security threats in a MANET will be mostly taken into account by identifying legitimate group members.

Once group member authentication has been completed in a MANET, the next phase is protection of each communication among group members. The use of a shared group key for message encryption and decryption fulfills the requirement; what is more critical and complex is the management and distribution of the group key. Some pertinent issues are; (1) legitimate members should be able to obtain the group key in a busy network where network topology and mobile nodes are dynamically changed, and (2) the group key should not be disclosed by non-members, even if the messages exchanged between a new member and other nodes are eavesdropped.

According to these observations, we propose a knowledge-based group member authentication protocol used in an infrastructure-less MANET. It consists of a **group member authentication** structure and a **secret group key management** structure. After the completion of both phases, the approved group member obtains a secret group key and can establish secure communication with legitimate group members.

The remainder of this paper is organized as follows: In Section II we investigate and analyze related work done by other researchers. In Section III we summarize *Zero Knowledge Proof (ZKP)* and *Threshold Cryptography* techniques. Afterwards, in Section IV we detail the communication model and structure of our knowledge-based authentication and its actual implementation. This section includes the explanation of share holder discovery, vector of knowledge, and public key encoding techniques that contribute feasibility to our protocol. Section V provides experimental results, and finally we conclude in Section VI by outlining several topics for future work that will lead to further improvements of our protocol.

II. Related Work

Regarding a group key sharing mechanism, Group Key Agreement (GKA) protocols such as Cliques [1][2] are viable candidates for group applications. The main contribution of GKA is to bypass the use of secure channel for new key redistribution (synchronization) to all members. In the group key agreement procedure, however, when any group member joins or leaves to and from MANETs, the previously agreed group key must be recalculated and retransmitted. This procedure has a major impact as nodes join and leave a group frequently.

In contrast to the Group Key Agreement protocols, the group key sharing mechanism based on *Threshold Cryptography* [3][4] is a very beneficial approach. In fact, there are several references that apply the key sharing technique for the MANET environment [5][6][7]. Some important properties of the MANET and possible solutions are discussed therein. However, there is still a big gap between what is reported as the state-of-the-art in the literature from what is implemented in practice.

For instance, the authors in [5] do not pursue several security goals (e.g., node authorization) and related issues even though such matters are of concern to certain applications. In [6] the authors provide services by distributing the Certified Authority (CA) functionality to each mobile node even though this kind of implementation cannot always be fulfilled in an arbitrary MANET setting. A more recent proposal [8] describes a more interesting architecture whose approach is similar to ours. The main contribution of their work adheres to an effective group key distribution mechanism among ad-hoc group members. However, the distribution of all group members' public keys – a necessary condition – may be a weak assumption when a large number of mobile nodes exist in the network. Collectively, all of these references are mainly concerned with mechanisms to control admission to a secure group, but less so with the interaction of group security in different ad-hoc networks. Furthermore, these protocols are either lacking a group member authentication phase or assume that it is carried out by some external functions which may not be feasible for the deployment.

III. Background Techniques

Our group member authentication protocol employs two building blocks; *Zero Knowledge Proof (ZKP)* and *Threshold Cryptography*.

A. Zero Knowledge Proof

The protocol ZKP, first introduced by [9][10], is an interactive or probabilistic proof and it demonstrates that one node (named "*prover*") has certain information without revealing the information to another node (called "*verifier*"). As shown in Fig.1, a typical round in ZKP consists of the following steps:

1) A *commitment* message is sent from *prover*.
2) A *challenge* is sent from *verifier*.
3) A *response* to the challenge from the *prover* is sent.
4) The protocol may be *repeated* for many rounds.
5) Based on the *prover*'s responses in all the rounds, the *verifier* decides whether to *accept or reject* the proof.

There are several well-suited algorithms for ZKP, like factorization of the product of large primes, graph isomorphism, and discrete logarithms. These algorithms are typical

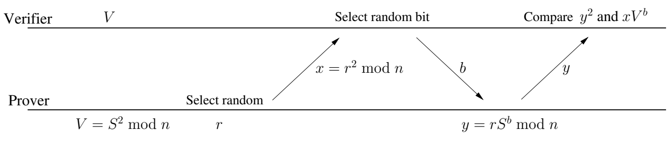

Fig. 1. ZKP procedure.

NP-complete (Nondeterministic Polynomial) problems. In the following, we will describe an algorithm based on factorization of the product of large primes that implements ZKP. This is the chosen algorithm in our protocol with further improvements as detailed in Section IV-E.

Here we have an arithmetic modulo N where $N = pq$, and p and q are large primes. Factoring N is assumed to be intractable. The verification procedure consists of rounds of interaction, which shows that the *prover* knows the square root of a published number without revealing any information about the value of the square root itself.

As shown in Fig.1, the *prover* first publishes the quadratic residue $V = S^2 \bmod N$, for which it claims to know the root S. When the *prover* wishes to prove its knowledge of S to the *verifier*, it runs several rounds of interaction. In each round, the *prover* chooses a new random number r and sends $x = r^2 \bmod N$ to the *verifier*. Now, the *verifier* chooses a random bit b, and sends it to the *prover*. The *prover* replies with $y = rS^b \bmod N$. To verify the *prover*'s claim, the *verifier* computes y^2 and compares it with xV^b.

In the above algorithm, only the *prover* can successfully complete the protocol for both possible values of b. This is clear since knowledge of r and rS implies that the *prover* knows S as well (i.e., $\frac{rS}{r} = S$). Note that in this approach the *prover* should never reuse an r; if an adversary collects all sets of both responses, it may be able to send replies to the *verifier* and possibly succeed in impersonating the *prover*.

B. Threshold Cryptography

We briefly review a novel cryptographical technique [3][4] that is used as part of our protocol. This technique is based on sharing secret among nodes. The idea comes from Shamir's discussion about the company's secret key [3].

An (n, t) threshold cryptography (where $n \geq t$) is defined as a scheme which allows n nodes to share the ability to perform a cryptographic operation, so that any t nodes can perform this operation jointly (i.e., *availability*), whereas it is infeasible for at most $t - 1$ nodes to do so (i.e.,*confidentiality*), even by collusion.

We use Shamir's scheme to generate the group key as well. Based on this scheme, in order to allow any t out of n nodes to construct a given secret, a $t-1$-degree polynomial is constructed such that the constant coefficient (i.e., S) is the secret and all other coefficients are random elements:

$$y = f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \cdots + a_1 x + S \quad (1)$$

In this case, each of the n shares is a pair of (x_i, y_i) of numbers such that $f(x_i) = y_i$ where $i \in \{1 \dots n\}$, $x_i \neq 0$. Now given any t shares, the polynomial is uniquely determined and hence the secret S can be computed by using Lagrange interpolation (where $l_i = \prod_{j=1,j\neq i}^{j=t} \frac{j}{j-i}$). However, given $t-1$ or fewer shares, the secret cannot be found.

In an (n,t) threshold cryptography scheme, we divide the secret group key S of the service into n shares $(s_1, s_2, \ldots, s_n)$, assigning one share to each node. We call $(s_1, s_2, \ldots, s_n)$ an (n,t) sharing of S.

IV. Protocol Design

A. Overview

Both of **group member authentication** and **secret group key management** are indispensable procedures for establishing the secure group communication in a MANET. To fulfill this requirement, we present an overview of our knowledge-based group member authentication protocol that employs the background techniques.

Firstly, we focus a **group member authentication** structure. While a secret group key is used only for encrypting and decrypting group communication in general, we apply it for identifying the group that a mobile node belongs to. In other words, each secret group key can be defined as a unique identifier; hence our protocol examines secret group keys held by a mobile node and then recognizes the criteria of group membership status of the node. In our protocol, a set of the secret group keys on a mobile node is called its "knowledge", and we recognize the node's knowledge shows all groups the node previously joined.

When a mobile node wants to become a new group member, the node looks for legitimate group members in the same network and tries to communicate with them. These legitimate group members then investigate the node's knowledge, compare the knowledge with pre-defined "required group membership," and evaluate whether the node can join the group.

In this knowledge verification procedure, an adversary must not be able to succeed to steal any meaningful information even if he eavesdrops all the information exchanged between the new node and group members. Following this line of thought, our protocol employs ZKP algorithm, which gives a method to verify a secret key without disclosure of any secure information. In ZKP, a new node behaves as *prover* and legitimate group members behave as *verifiers*. In a ZKP session, *verifier* does not need to use a secret key for the key verification; while the node's knowledge consists of secret group keys the node previously joined, the required group membership consists of publicly available "verification keys" corresponding to the secret group keys. The verification keys are equivalent to the quadratic residue V explained in Section III-A.

After the knowledge verification procedure is completed, the new node is ready to obtain the secret group key as the new group member. Here we can see that threshold cryptography is a beneficial approach as in a **secret group key management** structure of our protocol; the secret group key is divided to n shares and later generated by a new group member by the response of t group members (among n nodes). This proposal makes the protocol be robust, because it does not require a key server, and hence it works even within a busy MANET. With additional components, we also can reasonably escape from threats of disrupting the secret key generation by Verifiable Secret Sharing (VSS) [8] in order for legitimate member nodes to testify the validity of each share.

So far, when the node obtains a new secret group key, our protocol recognizes that the node joins the new group and increases its current group membership. In other words, the node's knowledge can be improved step by step when the node obtains different group keys in our proposed procedure.

According to the knowledge-based group member authentication, there are several assumptions in the target of our protocol. Our protocol does not protect the situation that an adversary invades (or cracks) a legitimate group member node and steals a secret group key (S) from the disk or memory on that node. And since a mobile node that has all the required group membership can become the group member (i.e. can obtain the group key) in our protocol scheme, our protocol neither protects attacks from such potential group member.

B. Entities

Before discussing about the detail communication model, we introduce three entities used in our protocol: (1) *new member*, (2) *share holder* and (3) *dealer*.

New member is a mobile node that tries to become a group member in a network. *New member* is initially a prospective group member, and later obtains shares from t *share holders* (in an (n,t) threshold cryptography) and generates a secret group key. It must know the verification key corresponding to the secret group key upon request.

Share holder is a set (n numbers) of legitimate group members. This entity holds own share of a secret group key and the required group membership (i.e. a list of verification keys). It verifies the knowledge of *new member* and gives the share of the group key to the valid node. *Share holder* behaves as *verifier* in ZKP.

In each group, one *dealer* exists as the initial entity. It appears only in a group initialization phase as mentioned in Section V-A.

C. Communication Model

Following steps describe the communication flow between *new member* and *share holder* as with Fig.2:

1) *New member* discovers IP addresses of *share holders* for a secret group key S by a share holder discovery procedure (detailed in Section IV-D).
2) *New member* sends the request to the selected *share holder* to retrieve share for S.
3) *Share holder* sends back acknowledge.
4) *New member* and *share holder* start *challenge* and *response* for a ZKP session to verify whether the *new member* fulfills the required group membership. *Share holder* obtains the *new member*'s public key (P_k) during this session (detailed in Section IV-E).
5) *Share holder* gives its share to the *new member* securely.
6) All above procedures are repeated t times with different *share holders*.

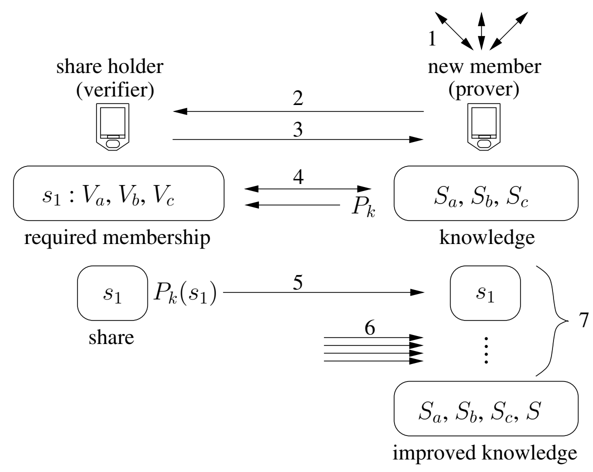

Fig. 2. Communication flow between *verifier* and *prover*.

7) *New member* finally generates S from t shares by threshold cryptography.

In each step from 2 to 5, our protocol defines "time out" value (e.g. 5 seconds) in order that *new member* cancels the procedure with selected *share holder*, because it may happen that the *share holder* leaves from the network within the communication. When *new member* cancels the procedure, it restarts communication with another *share holder* just after the time out.

D. Share Holder Discovery

A **share holder discovery** procedure enables *new member* to find *share holders*' addresses. This procedure is controlled by `SHARE_DISCOVERY` and `SHARE_REPLY` messages; when *new member* joins a group, it multicasts `SHARE-_DISCOVERY` message to the corresponding multicast address (explained next), and when the *share holder* listens this discovery message, it unicasts `SHARE_REPLY` message to the *new member* with the required group membership and t.

As the collaborative mechanism, an "expanding ring search" (sometimes called "TTL scoping") is inherited as part of the share holder discovery procedure. It helps the situation when *new member* cannot find t or more share holders within a single hop wireless link (i.e. TTL=1). In an expanding ring search, the `SHARE_DISCOVERY` message is forwarded by increasing its TTL as with $2, 3, \ldots,$ until the *new member* receives `SHARE_REPLY` messages from t or more *share holders*.

One aspect of this trait is that independent discovery messages that belong to different groups are possibly transmitted within the network, while *share holders* do not need to respond to different group members. Our protocol therefore arranges the destination multicast address used by a `SHARE-_DISCOVERY` message in order to filter out unneeded discovery messages on *share holder* side. More precisely, the corresponding multicast address for each `SHARE_DISCOVERY` message is generated with the following rule:

$$\{\textit{multicast address prefix}\} \mid \{\textit{leftmost } H(V)\}$$

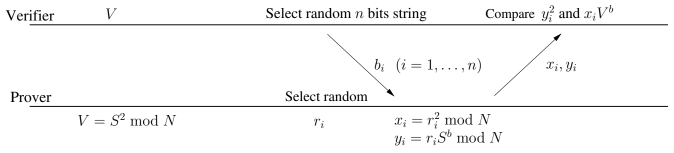

Fig. 3. Vector of knowledge.

where a *multicast address prefix* comes from a 16 bits (for IPv4) or 64 bits (for IPv6) address prefix[1], and *leftmost* $H(V)$ indicates a "group identifier" that contains the leftmost 16 bits (for IPv4) or 64 bits (for IPv6) of the SHA-1 [11] hash of the verification key (V) corresponding to the secret group key. Here, "|" is an operation to concatenate left and right strings. In this environment, we can relatively reduce the possibility of multicast address duplication between different group members. Hence *share holders* listening on the same multicast group address are usually the same group members, and *new member* expects that `SHARE_REPLY` messages will be sent from appropriate *share holders*. Note that, even if the multicast address is duplicated with other groups, *new member* can finally choose corresponding `SHARE_REPLY` messages by checking the verification key of the secret key in the message.

E. Vector of Knowledge and Share Encryption

Our protocol requires several (or many) ZKP sessions for the knowledge verification. This may introduce a large number of message exchange or long response for group member authentication and finally cause some performance drawback for the group key generation. But in fact ZKP can be implemented in a parallel fashion, making the public and private information be a set of quadratic residues modulo N. We can do as many rounds in parallel as the keys are available in the set. **Vector of knowledge** hence aims to speed up our protocol by reducing the number of ZKP messages (Fig.3).

After the ZKP completes, the next phase is the share transmission to the *new member*. However, if *share holder* sends its share over non-secure channel, malicious nodes may collect more than t shares and finally find the secret group key. The *share holder* may want to use an asymmetric key (e.g. a *new member*'s public key) to encrypt its share and transmit it to the *new member* securely. Yet, the protocol using an asymmetric key needs to verify the key owner to avoid threats such as a man-in-the-middle attack, whereas it is generally difficult in an infrastructure-less environment.

In our protocol, we propose a **public key encoding** scheme. This technique enables; (1) *new member* sends own RSA[2] public key [12] to *share holders* within a ZKP session, and (2) each *share holder* verifies that the key owner is the *new member* seamlessly.

Let us now detail the procedures of step 4 in Section IV-C with both of *vector of knowledge* and *share encryption* techniques:

[1]We tentatively assign one site-local address prefix for our protocol.

[2]RSA is a registered trademark or trademark of RSA Security Inc. in the United States and/or other countries.

a) When *share holder* receives a SHARE_REQUEST message (i.e. a *commitment* message), it randomly chooses bit strings and sends a vector of bit strings, $B_1 = \{b_{11}, b_{12}, \ldots, b_{1m}\}$, to *new member* with a SHARE-_CHALLENGE message. m is defined by *share holder* as rounds of ZKP's *challenge* (e.g. $m = 20$) and b_{1i} is the i-th bit of B_1.

b) *New member* computes a set of values $X_1 = \{x_{11}, x_{12}, \ldots, x_{1m}\}$ given by $x_{1i} = r_{1i}^2 \bmod N$, where r_{1i} is a random number and N is a very large prime number.

c) *New member* computes a set of values $Y_1 = \{y_{11}, y_{12}, \ldots, y_{1m}\}$ given by $y_{1i} = r_{1i} S_k^{b_{1i}} \bmod N$, where S_k is one of the new member's knowledge to be verified, like S_a in Fig.2.

d) *New member* computes $\mathrm{H}(Pk) = B_2 = \{b_{21}, b_{22}, \ldots, b_{2n}\}$. Here, H is a one way hash function (SHA-1 [11] is used in our protocol), Pk is a public key of the *new member*, B_2 is a set of bit values of the output of the hash function, and n is 160 according to SHA-1's output.

e) *New member* computes a set of values $X_2 = \{x_{21}, x_{22}, \ldots, x_{2n}\}$ given by $x_{2i} = r_{2i}^2 \bmod N$, and a set of values $Y_2 = \{y_{21}, y_{22}, \ldots, y_{2n}\}$ given by $y_{2i} = r_{2i} S_k^{b_{2i}} \bmod N$.

f) *New member* sends X_1, X_2, Y_1, Y_2 and Pk to *share holder* with a SHARE_RESPONSE message.

g) *Share holder* computes $\mathrm{H}(Pk)$ and verifies the values of Y_1 and Y_2 by comparing with y_{ji}^2 and $x_{ji} V^{b_{ji}}$ where $j \in \{1, 2\}$. If the *share holder* finds all y_{ji} are correct, it authenticates the *new member* and recognizes the Pk is the *new member*'s public key. If anything fails, *share holder* rejects the *new member*.

h) *New member* and *share holder* repeat above procedures for all the required group membership. If the different Pk is sent within the repetitions, *share holder* rejects the *new member*.

When all above procedures are completed, *share holder* encrypts own share with given Pk and sends it to the *new member* with a SHARE_DONE message. The *new member* then obtains t shares from t *share holders* securely.

These procedures fulfill the demand in which (1) *new member* verifies that it has all the required group membership, and (2) *new member* can successfully send own Pk, because only the node that has corresponding secret group key can prove X_2, which is given by $\mathrm{H}(Pk)$ and B_2.

V. Experimental Results

Our protocol is implemented in Java[3], using Java 2 SDK 1.4.2_05. The source code currently consists of approximately 40 classes. We have adopted Triple DES (24 bytes) as the cryptographic function with embedded Java class, `javax.crypto.KeyGenerator`.

[3]Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

For all tests described below, we have prepared 2.40 GHz Pentium[4] 4-based laptops with 192MB memory running Fedora[5] Core 2 Linux[6] and 2.80 GHz Pentium-4 laptops with 240MB memory running Windows[7] XP. Note that, to measure the protocol performance without dependency of an immature IBSS mode driver implementation, we needed to connect each machine through 802.11b BSS mode in a dedicated wireless network.

A. Setup

A bootstrapping phase consists of (1) group initialization and (2) node initialization steps for our protocol setup.

A group initialization includes the required group membership configuration, a secret group key initialization, and an (n, t) definition for threshold cryptography. *Dealer* keeps these configurations, computes a polynomial for threshold cryptography, and distributes shares and the corresponding required group membership to n *share holders*. After the end of this phase, *dealer* terminates its task from the network, removes all information related to the last shared key generation mechanism, and becomes an ordinary *share holder*.

A node initialization procedure – which is usually done just after a node is initially booted – is required only once for each node. Preparing the "initial knowledge" is its main task. Afterwards a node subsequently uses this initial knowledge and improves knowledge at a later time. In our current protocol, we need to assume that the initial knowledge must be issued by a company that a user of the node is working, or by an ISP that the node usually connects.

B. Evaluations

Figure 4 shows the average time of secret group key acquisition from t different *share holders* in an (n, t) threshold cryptography environment. This procedure includes the knowledge verification with t *share holders* by ZKP so as to complete the whole story explained in Section IV-C. All tests have been done with the following set of parameters:

- 1024 bits modulo N
- Number of required knowledge: 5, 10, 20 and 30
- Number of knowledge held on *new member*: 35
- Bit strings length for ZKP: 20
- RSA public key length: 1024 bits

According to this experiment, while many rounds in the ZKP procedure may slightly take longer response, the vector of knowledge implementation might minimize its performance impact. In fact, based on our analysis, the ZKP procedure takes less than 10% of the total response time, yet data transmission over wireless link possesses the highest ratio.

Now measuring the message size of the protocol is additional important criteria. Figure 5 shows the total message size

[4]Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

[5]The Fedora trademark is a trademark of Red Hat, Inc. (Red Hat) in the United States and other countries.

[6]Linux is a registered trademark of Linus Torvalds.

[7]Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

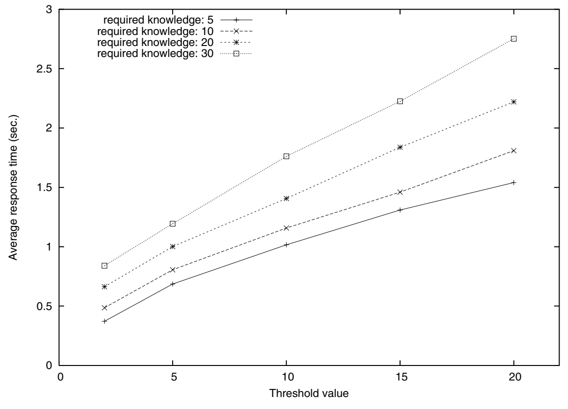

Fig. 4. Average time to become a group member.

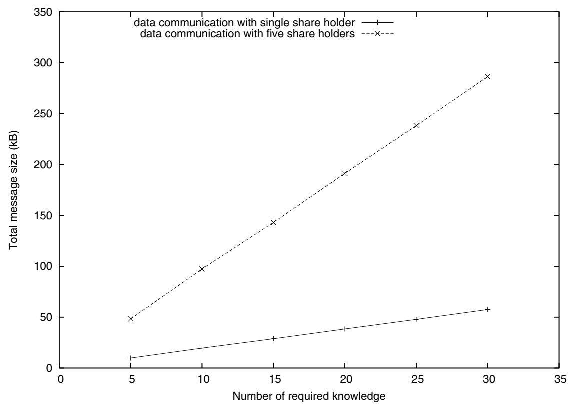

Fig. 5. Data communication cost to complete the protocol.

(including *new member*'s public key) transmitted in the secret key acquisition procedure. We could then estimate the data communication cost of our protocol. Actually the bit strings of ZKP accounts for approximately 90% of the total message size, and it would be proportionally increased to the threshold value. Reducing rounds of ZKP (e.g. 20 in our test) will clearly reduce the total message size, but trade-off exists between the protocol robustness and performance.

VI. Conclusion and Future Work

From a technical perspective, the main contribution of this paper is in analyzing secure group communication in a MANET and presenting the group member authentication protocol and its actual implementation. Our protocol uses *Zero Knowledge Proof* for group membership verification, and *Threshold Cryptography* for managing a secret group key in MANETs. Our protocol implementation shows the possible communication model and relatively reasonable performance, and therefore this work would give opportunities for making significant and realistic contributions in the real use of secure group communication in MANETs.

We have already identified our future work. It is related to the definition of n and t values in threshold cryptography. In our current protocol, these values should be configured by operation. In a feasible sense, these values will be related to the size of network or the number of group members and should be seamlessly adjusted in consequence of the condition. Furthermore the decision regarding which group member should work as *share holder* also belongs to the same thought. We will investigate an effective mechanism to address these issues.

One of the open issues is related to key revocation. In our current protocol the secret group key is not compulsorily revoked. It is because the group key is used as knowledge in our protocol and the meaning of knowledge should be kept for a long span of time. However, developing some intelligent and scalable group key revocation mechanism that keeps consistency of our knowledge-based protocol may be needed in future.

References

[1] M. Steiner, G. Ateniese and G. Tsudik, "New multiparty authentication services and key agreement protocols", IEEE Journal of Selected Areas in Communications, 18(4):1-13, 2000.

[2] G. Tsudik, M. Steiner and M. Waidner, "Key agreement in dynamic peer groups", IEEE Transactions on Parallel and Distributed Systems, August 2000.

[3] A. Shamir, "How to Share a Secret", Communications of the ACM, vol.22, pp.612-613, November 1979.

[4] Y. Desmedt, "Some Recent Research Aspects of Threshold Cryptography", Proc. Information Security, (Lecture Notes in Computer Science 1396), pp.158-173, Springer-Verlag, 1997.

[5] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks", IEEE Network Magazine, vol.13, no.6, Nov/Dec 1999.

[6] J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks", Proc. IEEE ICNP, November 2001.

[7] J. Hubaux, L. Buttyan and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks", Proc. ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), October 2001.

[8] M. Narasimha, G. Tsudik and J. Hyun, "On the Utility of Distributed Cryptography in P2P and MANETs", Proc. IEEE ICNP, November 2003.

[9] U. Feige, A. Fiat and A. Shamir, "Zero knowledge proofs of identity", Proc. the 19th ACM Symp. on Theory of Computing, pp.210-217, May 1987.

[10] S. Goldwasser, S. Micali, "The knowledge complexity of interactive proof systems", SIAM Journal on Computing, pp.186, 1991.

[11] D. Eastlake, 3rd and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC3174, September 2001.

[12] RSA Security, <http://www.rsasecurity.com/rsalabs/node.asp?id=2214>.