On the Utility of Distributed Cryptography in P2P and MANETs: the Case of Membership Control

Maithili Narasimha, Gene Tsudik, Jeong Hyun Yi*
School of Information and Computer Science
University of California at Irvine
{mnarasim,gts,jhyi}@ics.uci.edu

Abstract

Peer-to-peer systems enable efficient resource aggregation and are inherently scalable since they do not depend on any centralized authority. However, lack of a centralized authority, prompts many security-related challenges. Providing efficient security services in these systems is an active research topic which is receiving much attention in the security research community.

In this paper, we explore the use of threshold cryptography in peer-to-peer settings (both Internet- and MANET-based) to provide, in a robust and fault tolerant fashion, security services such as authentication, certificate issuance and access control. Threshold cryptography provides high availability by distributing trust throughout the group and is, therefore, an attractive solution for secure peer-groups. At least, so it seems... Our work investigates the applicability of threshold cryptography for membership control in peer-to-peer systems. In the process, we discover that one interesting recently proposed scheme contains an unfortunate (yet serious) flaw. We then present an alternative solution and its performance measurements. More importantly, our preliminary work casts a certain degree of skepticism on the practicality and even viability of using (seemingly attractive) threshold cryptography in certain peer-to-peer settings.

1 Introduction

Peer-to-peer (P2P) systems typically reside at the "edges" of the Internet or in mobile ad-hoc networks (MANETs) and are characterized by their ability to perform a task using distributed resources in a decentralized manner. In such settings, lack of any centralized authority results in formidable challenges as far as providing effective and efficient security services. Moreover, the dynamic nature of peer group membership adds a further complication. Security services need to be provided in an efficient and scalable manner while allowing for membership events that change the topology (i.e., the composition) of a peer group. In other words, the volatility of group membership, coupled with the absence of centralized authority, necessitate the distribution of highly sensitive operations, such as the functions of a Certification Authority (CA), throughout the peer group itself to ensure their availability.

Informally, *Threshold Cryptography* [4] distributes the ability to provide a cryptographic service such as decryption or signing. It offers better fault tolerance than non-threshold cryptography: even if some nodes are unavailable, others can still perform the task. Threshold cryptography also offers better security since no single entity is entrusted to perform the task in its entirety. Consequently, it seems like an ideal choice to provide security services, such as authentication and access control, in P2P systems.

More formally, threshold cryptography provides an elegant means to share critical functions among group members, in a way that a coalition of cooperating parties can jointly perform the function. Specifically, an (t, n) threshold cryptographic scheme allows n parties to share the ability to perform a cryptographic operation (e.g., decrypt or sign a message), in a way that any t parties can perform this operation jointly, whereas, no coalition of $(t-1)$ or fewer parties can perform the same operation. In this paper, we explore the applicability of current threshold signature schemes to providing security services for peer groups. We use group access control (more precisely, group admission) as an example to demonstrate and discuss the usefulness of threshold schemes. As discussed in section below, our work is not the first to explore this topic. (Interested readers are referred to [29] and [18] for further background and motivating factors.)

Contributions in Brief: the main contribution of this paper is in presenting and analyzing two solutions, both using threshold cryptography, for the problem of *membership*

[*] Author names appear in alphabetical order.

control. The first solution, based on so-called threshold RSA signatures, is a simple extension of the scheme proposed by Kong, et al.[17, 18, 16]. As discussed later in the paper, this method does not provide an important property known as *verifiability* and, hence, can not be used in a setting where malicious insiders (group members) can exist. Furthermore, it requires a trusted third party to initialize the group. The second solution, is based on threshold DSA signatures, overcomes these problems. Another contribution of our work is in demonstrating (via experimental results) the costs and performance of the two threshold schemes. Finally, this work also raises some important issues that question the practicality of using threshold cryptography in certain P2P settings where group membership is dynamic and group connectivity is sporadic.

Scope and Limitations: Group access control includes many issues, including membership control mechanisms and the more general issue of group security policy. This paper is concerned only with the mechanisms to control admission to a secure peer-group and does not address the specification and negotiation of group security policy. In the following, we assume the existence of such a policy. Furthermore, in an effort to keep our discussion general, we do not consider the impact of the underlying physical-layer characteristics of the peer-group network. Naturally, a wireless network will have additional security and performance concerns, as compared to its wired counterpart. However, these issues, unique to a wireless network, are beyond the scope of this work. Additionally, we are using a certificate-based approach to provide membership control. A common problem associated with such approaches is the revocation of certificates. However, although we acknowledge this issue and appreciate its difficulty, it is out of scope of this paper.

Organization: This paper is organized as follows: after addressing the related work, we describe (in Section 3) the particulars of membership control in peer groups and identify some design challenges. Section 4 overviews our generic membership control protocol. Next, Section 5 introduces necessary cryptographic building blocks. Then, in Section 6, we apply a threshold RSA based scheme for membership control and point out some of its shortcomings. In Section 7, we propose an alternative DSA-based scheme. Finally, Section 9 describes the implementation and the performance of the two schemes.

2 Related Work

Our work pairs threshold cryptography with peer-group security in an attempt to facilitate membership control mechanisms. In this section, we briefly examine relevant prior work from both threshold cryptography as well as peer-group security.

Threshold signatures are part of the general area known as threshold cryptography [3, 4]. Many flavors of threshold signature schemes are described in the research literature. Some schemes obtain robustness by providing *verifiable secret sharing* [2, 9]. Others periodically update the participants' secret shares via the technique known as *proactive secret sharing* [8, 12]. Some of these require a trusted centralized authority (dealer) to bootstrap the secret-sharing procedure, while others provide *joint secret sharing* and do not require any trusted authorities.

One of the earliest attempts to use threshold cryptography in a peer group setting is the work of Zhou and Haas [29] which proposed using threshold signatures to improve security in MANETs. They observed that using threshold cryptography to distribute the group signing key prevents one (or several) compromised node(s) from signing messages on behalf of the group.

More recently, Kong, et al. [17] used a similar approach to provide a scalable distributed authentication service in MANETs. They constructed a threshold RSA signature scheme (discussed below in section 6) that supports dynamic groups. Unfortunately, as will be shown below, this scheme does not provide an important property – verifiability. In a follow-on work, Luo, et al. [18] enhanced [17] to provide proactive secret sharing. In this paper, we adhere, to a large extent, to the design guidelines highlighted in [18].

3 Architecture

We begin by outlining some basic features and requirements for membership control in peer groups. (See [15] for further background on this topic.)

A group membership control mechanism must guarantee that *bona fide* members of a peer group have been "approved" to join that group. In other words, a membership control mechanism ensures that only those entities that have satisfied certain admission requirements are allowed membership. Admission requirements are clearly group-specific and form an integral part of group security policy. A prospective member needs to learn these admission rules, which triggers the need to codify such rules in a readily available document. We refer to this document as the *group charter* [15].

In addition to the group charter, a well-defined procedure for admitting a new member is needed. The simplest way to admit a new member is to enumerate all potential group members *a priori* via a (signed) public Access Control List (ACL). This method is ideally suited for a static group, where the information about all prospective group members is known in advance. However, in a dynamic group, such as the kind we are considering, membership might be impossible to enumerate.

An alternative way is to appoint a trusted *Group Authority* [15] to handle admission procedures. Having a single trusted authority, however, not only violates the peer nature of the group, but also introduces a single point of failure (and attack) and limits scalability. Although the Group Authority can be replicated for better availability, the scalability can not be addressed by replication alone. Furthermore, unpredictable network faults and partitions complicate placement of Group Authority "replicas" in the network. Our approach is to let the group members themselves to handle admission procedures. Concretely, any subset of at least t members (where t is a threshold) can jointly decide to admit a new member to their group. This distribution of responsibility for member admission among all group members is clearly in line with the peer spirit of the group where all members have equal rights and duties. It also ensures that any t (or more) members can provide requested service. The decision to admit a prospective member is made via a voting protocol which is based on a (t, n)-threshold signature scheme where n $(n \geq \ t)$ is the current group size. Details of this protocol are presented in the subsequent sections. The high-level description is as follows:

1. Every group has a group public key PK and a corresponding private key SK which are used primarily for verifying and generating signatures, respectively.
2. The group secret key SK is securely shared among all group members using a (t, n) threshold scheme; each member has her own unique secret share.
3. Using her secret share, a member can generate a partial signature on any message.
4. In a (t, n) threshold scheme, any subset of at least t current members can "pool" their partial signatures to produce a valid signature (with the group signing key SK) on any message.
5. Using this feature, a quorum of t members can effectively sign on behalf of the group to generate a *Group Membership Certificate* (GMC) for a new member.

   Note that a signed membership certificate is insufficient to fully admit a member: the new member must also be issued her own (new) secret share which would allow her, hereafter, to take part in admitting future members. In other words, an (t, n) scheme must be dynamically adjusted to become (t, $n + 1$) scheme.

3.1 Architectural Details

Membership dynamics: Group membership dynamics can influence the basic design of membership control mechanisms. As stated earlier, we consider dynamic groups in which any member may join and leave the group over time.

Threshold and group size: We distinguish between fixed and dynamic thresholds. A threshold may be specified as either the minimum number of votes (say, t) or as a fraction of the current group size depending on the group security policy. A fixed threshold is essentially a t-out-of-n model where t is fixed and n (current group size) may vary over time. In contrast, in a dynamic threshold setting, t increases or shrinks in proportion to n.

With the fixed threshold policy, the minimum number of votes (t) required for admission is constant throughout the group's lifetime. However, fixing a static value t for a dynamic group at the time of group formation may not be efficient. For instance, assume that an adversary manages to *permanently* compromise t group members. In this situation, refreshing individual members' secret shares using a proactive secret sharing scheme is not always sufficient. In this case, proactive secret sharing does not help if the resultant size of the group (excluding t compromised members) is less than t, i.e., $(n - t) \leq t$. In such cases it is necessary to reduce the threshold t in order for the group to operate. A similar situation can also arise when a large number of members leave the group, resulting in a new group size being less than t. Similar issues arise at group inception time when first few (fewer than t) members join the group. In such cases, special admission rules are needed. On the other hand, with the dynamic threshold, the minimum number of votes is a fraction of the number of current group members. The main problem here is the need to securely and reliably determine the number of current group members, which, unfortunately, turns out to be a major problem in practice. Techniques to increase or decrease the threshold form a core part of the peer group admission mechanism.

Determining group size: In order to support a dynamic threshold (where threshold size t is a fraction of the current group size n), it is necessary to securely and reliably determine the current group size. This is a challenging problem, especially, in a completely distributed, asynchronous and decentralized dynamic group setting. One trivial way to solve it is by imposing or assuming a trusted authority charged with maintaining up-to-date membership information. Every member is then required to periodically send a *heart-beat* message to this trusted authority which aids in maintaining the current group size. Clearly, the "peer" nature of the group is violated by the presence of the trusted authority. Nonetheless, we stress that the authority is only trusted to keep track of the membership information and not the actual group secret. We recognize that a trusted authority can represent a single point of failure and an adversary can launch denial-of-service attacks against it and disrupt service. Although replication can be used to reduce exposure and improve availability of the trusted authority, some environments (e.g. highly volatile MANETs) would not fit this model.

Need for proactive secret sharing: In traditional threshold secret sharing, an adversary needs to compromise t entities in order to expose the secret. Gradual break-ins into t entities over a long period of time might be possible since in normal secret sharing, the secret shares that have been distributed remain unchanged. Therefore, traditional threshold secret sharing is not sufficient for long-lived secrets. This prompts the need to refresh individual secret shares, periodically, without changing the existing group secret SK. To this end, *proactive secret sharing* mechanisms refresh individual secret shares under such constraints.

Secret share acquisition: A newly joining member must acquire a secret share of the group secret SK for herself. This enables her to participate in future voting procedures in order to admit other new members. The secret share acquisition also needs to be carried out in a distributed manner. The newly joining member should be able to receive t partial secret shares from each of the t members that elect to admit her to the group. The new member can then securely combine these partial shares to obtain her own secret share of the group secret key SK.

Verifiable secret shares: If there are malicious secret share holders in the group, values other than the actual shares may be used by these members to generate false partial signatures or partial secret shares. Thus, when a prospective member gets a partial signature (or a partial secret share) from a current member, a verification mechanism to check if the received value is valid must be provided.

PKC based approach for identity binding: Possession of a GMC does not prove that the GMC actually belongs to the bearer. There needs to be some binding information that asserts ownership to the bearer. One way to accomplish this is by requiring for every group member to have a standard X.509 public key certificate (PKC) issued by a recognized Certification Authority (CA) [13]. The GMC simply needs to incorporate the public key of the member from her PKC. Now the member (bearer of a GMC) can prove ownership of the GMC by demonstrating knowledge (e.g., by signing a message) of the private key corresponding to the public key referred to in in the GMC.

3.2 Notation

The notation used in this paper is summarized below:

t	threshold
TD	trusted dealer
GID	group identity
M_i	group member i
SK_i,PK_i	secret key and public key of M_i
$SIG_i(x)$	signature of message x generated with SK_i
ss_i	secret share of M_i
$pss_j(i)$	partial secret share for M_i by M_j
SL_i	signer list to generate M_i's signature
PKC_i	public key certificate of M_i
GMC_i	group membership certificate of M_i

4 Membership Control Protocol

In this section, we describe the general group membership protocol. At the end of the protocol, given that enough members ($\geq t$) approve admission, the prospective member who initiated the protocol becomes a member of the group and obtains her own membership certificate GMC_{new}.

Step 0. Setup: In the initial (bootstrapping) phase, each group member M_i obtains her secret share ss_i and a membership certificate GMC_i from a centralized dealer or by collaborative computation among initial group members. For example, the threshold RSA scheme described in section 6.1 only supports share distribution by a trusted dealer, while the threshold DSA scheme (section 7.1) provides share distribution by both the dealer and the group founders. In both schemes, after initializing t group founders, the centralized dealer is no longer needed.

Step 1. Join Request: A prospective member M_{new} initiates the protocol by sending a JOIN_REQ message to the group. This message is signed by M_{new} and contains, among other values, M_{new}'s public key certificate (PKC_{new}) and the target group name. How this request is sent to the group is application-dependent. Note also that M_{new}'s certificate does not have to be an identity certificate (i.e., PKC_{new}); it could as well be a group membership certificate for another group.

Step 2. Voting: Upon receipt of the JOIN_REQ, a group member first extracts the sender's PKC_{new} and verifies the signature. If a voting member approves of admission she replies to M_{new} with a signed (and well-typed) message. Threshold signature schemes in sections 6 and 7 are applied for this step.

Step 3. GMC Issuance: Who issues the GMC_{new} for M_{new} depends on the security policy and accordingly has several choices. In the fully distributed setting that we are considering, once enough votes are collected, M_{new} verifies the individual votes, and computes her own GMC_{new}.

Step 4. Share Acquisition: If M_{new} becomes a legitimate member, she needs to obtain her own secret share ss_{new} which enables her to participate in future admission protocols. M_{new} obtains ss_{new} by combining partial secret shares generated by a group of t members in a distributed manner. We emphasize that these shares, which are supplied by current members, must be verified because some of the current members can be malicious.

5 Cryptographic Primitives

In this section, we briefly describe various cryptographic techniques underlying threshold RSA and DSA signature schemes. We assume the reader is familiar with the basics of plain (i.e., non-threshold) RSA [25] and DSA [21] signature schemes.

5.1 Secret Sharing

The idea of secret sharing [20] is to divide a secret S into pieces or $shares$ which are distributed amongst users such that pooled shares of certain subsets of users allow reconstruction of the secret S. We use Shamir's secret sharing scheme [26] which is based on polynomial interpolation. To distribute shares among n users, a trusted dealer chooses a large prime q, and selects a polynomial $f(z)$ over $\mathbb{Z}_q$ of degree $t-1$ such that $f(0) = S$. The dealer computes each user's share ss_i such that $ss_i = f(i) \mod q$, and securely transfers ss_i to user M_i. Then, any group of t members who have their shares can recover the secret using the Lagrange interpolation formula: $f(z) = \sum_{i=1}^{t} ss_i\, l_i(z)$ $\pmod q$, where $l_i(z) = \prod_{j=1,j\neq i}^{t} \frac{z-j}{i-j} \pmod q$. Since $f(0) = S$, the shared secret may be expressed as:$S = f(0) = \sum_{i=1}^{t} ss_i\, l_i(0) \pmod q$.

Thus, S can be recovered only if at least t shares are combined. In other words, no coalition of less than t members yields any information about S. The $l_i(0)$ are non-secret constants, and may be precomputed.

5.2 Joint Secret Sharing (JSS)

This scheme (due to Pederson [23]) extends Shamir's secret sharing by removing the need for a centralized dealer to choose a polynomial and distribute shares. In this scheme, the group members collectively choose shares corresponding to Shamir's secret sharing of a random value without the dealer. The main idea here is that the polynomial itself is shared such that $f(z) = f_1(z) + \cdots + f_n(z)$, where $f_i(z)$ is the polynomial of member M_i over $\mathbb{Z}_q$.

Suppose there are n members in a group $(M_1, \cdots, M_n)$. It will be assumed that the members of the group have previously agreed on the prime q.

1. Each M_i chooses at random a polynomial $f_i(z) \in \mathbb{Z}_q$ of degree $t-1$ such that $f_i(0) = S_i$. Let $f_i(z) = f_{i0} + f_{i1}z + \cdots + f_{i,t-1}z^{t-1} \pmod q$, where $f_{i0} = S_i$.
2. M_i computes M_j's share $ss_j^i = f_i(j)$ for M_j $(j = 1, \cdots, n)$, and *securely* sends it to M_j (in particular M_i keeps ss_i^i). Note that the transmission of share values should be done through the secure channel.
3. M_j computes her share ss_j of the secret S as the sum of all shares received; $ss_j = \sum_{j=1}^{n} ss_j^i$

Let f denote the group polynomial over $\mathbb{Z}_q$. It is given by: $f(z) = f_1(z) + \cdots + f_n(z)$. By construction $ss_i = f(i)$, $i = 1, \cdots, n$, and therefore ss_i is a share of $S = f(0) = (= \sum_{i=1}^{n} S_i = \sum_{i=1}^{n} f_{i0})$. Once again, any coalition of t members can jointly recover the secret S using Lagrange interpolation as in basic Shamir's secret sharing [26].

5.2.1 Joint Zero Secret Sharing

This is a variant of joint secret sharing where the shared secret is zero. It is used in *proactive* secret sharing [12] for re-randomizing a secret share. In other words, this scheme can be used to change the individual secret shares of the users without changing the group secret. It can be achieved, quite simply, as follows: suppose $f(z)$ is the original polynomial such that $f(0) = S$, and $g(z)$ is another polynomial such that $g(0) = 0$. The interpolation of shares by addition of two polynomials, $h(z) = f(z) + g(z)$, provides the same secret S since $h(0) = f(0) + g(0) = S$. We note that proactive secret sharing assumes that members follow the protocol *faithfully* and destroy (or erase) their previous shares as soon as they obtain new ones.

5.3 Verifiable Secret Sharing

If we suppose that some group members can become malicious or compromised by an adversary, they may attempt to "cheat" by using incorrect secret shares in order to deny/disrupt the service. To remedy the situation, a more advanced technique, *Verifiable Secret Sharing* (VSS) [7] can be used. It basically provides a means to detect incorrect secret shares.

To be more specific, VSS setup involves two large primes p and q, and an element $g \in \mathbb{Z}_p^*$ chosen in a way that q divides $p-1$ and g is an element of $\mathbb{Z}_p^*$ which has order q. The procedure for the trusted dealer to distribute the shares is the same as in section 5.1. VSS is achieved by the following procedure:

Witness generation: The trusted dealer TD randomly selects a polynomial $f(z)$ over $\mathbb{Z}_q$ such that $f(z) = a_0 + a_1z + \cdots + a_{t-1}z^{t-1} \pmod q$, computes secret shares ss_i, and transfers them to each user securely. Also, TD chooses an element $g \in \mathbb{Z}_p^*$ of order q, and computes w_i $(i = 0, \cdots, t-1)$, called the **witness**, such that $w_i = g^{a_i}$ $\pmod p$. Then, TD publishes these w_i-s in some public domain (e.g., a directory server)[1].

Share verification: When t members receive their share ss_i, each member M_i verifies ss_i by checking that

$$g^{ss_i} \overset{?}{=} \prod_{j=0}^{t-1} (w_j)^{i^j} \pmod p$$

5.4 Partial Share Generation

A new member M_{new} receives t shares $pss_j(new)$-s from a group of t members. It will be assumed that the t number of indices, $j(= 1, \cdots, t)$, are given to each M_j by M_{new}. The details are as follows: Each M_j computes a partial secret share for M_{new} as: $pss_j(new) = ss_j \cdot l_j(new)$

[1] In case of JSS, where the group polynomial is jointly selected by the members, this step is carried out by each of the members individually

$\pmod p$, where ss_j is M_j's own secret share. Then, M_j securely sends $pss_j(new)$ to M_{new}.

Note that Lagrange coefficients $l_j(new)$ are *publicly known*, and therefore, M_{new} can derive ss_j knowing $pss_j(new)$. In order to prevent this, the *shuffling* technique proposed in [17] should be used. However, as a consequence of using this technique, the partial secret shares cannot be individually verified. VSS can be employed only to check the correctness of the final sum of the partial shares.

6 Threshold RSA (TS-RSA)

Recently, Kong, et al. [17, 18, 16] proposed the use of threshold signature schemes for distributing the functions of a certification authority throughout a MANET. They suggested an RSA threshold signature scheme. We will refer to it as TS-RSA. In this section, we describe their scheme, apply it to membership control in P2P systems and analyze its security.

6.1 Setup

During initialization, the TD is involved in choosing a secret polynomial and distributing the shares. TD's presence is only required during this bootstrapping phase to initialize the first t members (t is the threshold). TD generates an RSA private key d which forms the group secret key SK and randomly selects a polynomial $f(z)$ over $\mathbb{Z}_N$ of degree $t-1$ such that the group secret SK is $f(0) = d$; $f(z) = d + a_1 z + \cdots + a_{t-1} z^{t-1} \pmod N$ where N is RSA modulus.

TD computes the witnesses (refer to section 5.3) and the secret shares for each member M_i $(i = 1, 2, \ldots, t)$ as follows: $ss_i = f(i) \pmod N$. In addition, TD issues M_i's membership certificate GMC_i signed with the group secret key d, and distributes it with ss_i.

6.2 GMC Issuance

A new member M_{new} can obtain her GMC_{new} with the consent of at least t existing members. Figure 1 shows the procedure of certificate issuance. The protocol consists of four rounds of communication.

$M_{new} \to M_i$:	$SIG_{new}(PKC_{new}, GID, etc.)$	(1)
$M_{new} \leftarrow M_j$:	$SIG_j(GMC_j)$	(2)
$M_{new} \to M_j$:	$SL_{new},\ GMC\_REQ_{new}$	(3)
$M_{new} \leftarrow M_j$:	$m^{d_j} \pmod N$	(4)

Figure 1. TS-RSA: Certificate Issue

1. The M_{new} sends a signed JOIN_REQ message which contains her public key certificate PKC_{new}, group name, etc. to at least t current group members $M_i(|i| \geq t)$.
2. Group members, who are willing to participate in the voting, reply with their respective membership certificates GMC_i to M_{new} after verifying PKC_{new}[2].
3. M_{new} picks, at random, t responding $GMC_j(j \in_R i, |j| = t)$, and constructs a signer list SL_{new} with the ID-s of selected members[3]. Then, M_{new} sends SL_{new} to each M_j, along with GMC_REQ_{new} which contains the content of GMC_{new}. The reason why we need to collect the signer list is that each M_j should know the index of t members (co-signers) in advance to compute the Lagrange coefficient for the partial signatures and shares.
4. Each M_j computes her own partial secret key, d_j by multiplying her ss_j with $l_j(0)$. Then, M_j computes the partial signature $m^{d_j} \mod N$ where m is the certificate body derived from GMC_REQ_{new} and sends it to M_{new}.
5. M_{new} *combines* these partial signatures $m^{d_j} \pmod N$ to generate the original signature $m^d \pmod N$. (Please note that simply multiplying the partial signatures will not give the actual signature, i.e., $\prod_{j=1}^{t} m^{d_j} \neq m^d \pmod N$). In order to obtain the true signature, the *t-bounded coalition offsetting* algorithm given in [17] must be used. (Please refer to [17] for details.) Finally, M_{new} gets her GMC_{new}.

6.3 Share Acquisition

When M_{new} becomes a member of the group, she needs her own secret share ss_{new}, in addition to the membership certificate GMC_{new}. The new member acquires the secret share in the following manner: (Note: This procedure, in reality, is combined with the step (4) of section 6.2)

Each M_j calculates a partial secret share $pss_j(new)$ as follows: $pss_j(new) = ss_j \cdot l_j(new) \pmod N$.

M_j *shuffles* $pss_j(new)$ (refer to section 5.4 for details) and sends the resulting value securely to M_{new}. Then, M_{new} can obtain her secret share ss_{new}, by simply summing up t partial secret shares, $pss_j(new)$ since $\sum pss_j(new) = ss_{new} \pmod N$.

6.3.1 A Security Problem

After summing up all $pss_j(new)$ values to obtain ss_{new}, M_{new} must check if ss_{new} is correctly computed. Unfortunately, the protocol **cannot guarantee** that each ss_{new} is correct, for the following reason:

Since ss_{new} is computed modulo N and not $\phi(N)$, it is impossible to verify the secret share using publicly known

[2] In order to secure the protocol against common attcks such as *replay, impersonation* and *interleaving* attacks [20], we note that it is necessary to include additional information such as timestamps and identity information of the sender as well as the receiver. However, in order to keep the protocol description simple, we omit these values.

[3] M_{new} obtains member ID-s from their respective GMC-s

witnesses (refer to section 6.3). The value of $\phi(N)$ is known only to the dealer during group initialization and destroyed thereafter. Obviously, group members *must not* know the value of $\phi(N)$. Therefore, it appears that this scheme does not provide verifiability of secret shares. In other words,

$$g^{ss_{new}} \neq \prod_{i=0}^{t-1}(w_i)^{new^i} \pmod{N}.$$

We now provide a trivial example to illustrate the problem. Let us assume that the secret polynomial is $f(z) = 77 + 2z + 5z^2 \pmod{119}$, where $N = 119$ the product of two primes: 7 and 17, and $g = 3$. (Note that the degree of the polynomial is 2, hence, the threshold $t = 3$). The witnesses of $f(z)$, which are *publicly* known, are as follows: $w_0 = 3^{77} = 12$, $w_1 = 3^2 = 9$, and $w_2 = 3^5 = 5 \pmod{119}$. Suppose a new member M_7 receives the following partial shares from t existing members M_2, M_3, and M_5: $pss_2(7) = 71$, $pss_3(7) = 74$, and $pss_5(7) = 72 \pmod{119}$. M_7 computes her share $ss_7 = pss_2(7) + pss_3(7) + pss_5(7) = 98 \pmod{119}$. To check verifiability of the secret share, she computes

$$g^{ss_7} = 3^{98} = 9 \pmod{119}.$$

Also, using the witness values, M_7 can get the right hand side of the verifiability checking equation (given above) as follows:

$$(w_o)(w_1)^7(w_2)^{7^2} = 1 \pmod{119}.$$

Therefore, even though ss_7 is correctly computed,

$$g^{ss_7} \neq \prod_{i=0}^{2}(w_i)^{7^i} \pmod{119}.$$

6.4 Summary

As illustrated above, the otherwise elegant scheme by Kong, et al. [17, 18, 16] fails to provide verifiability of partial signatures as well as partial secret shares. As a result, malicious or compromised users can send fake shares to new members without being detected. This limits the scheme's applicability in providing security services in a dynamic group setting. Another drawback of this scheme (albeit, a relatively minor one) is that it requires a trusted dealer to generate the group secret (RSA key) and share it among the initial members. Although presence of the dealer is limited to a short period of time (bootstrapping), assuming such a trusted authority may not be feasible in certain application scenarios.

Other schemes have been proposed in the cryptographic literature. Notably, Gennaro, et al. [9] and Shoup [27] proposed threshold RSA signature schemes that provide verifiability. However, their schemes still require a trusted dealer to generate the RSA keys. Boneh and Franklin [1] developed a method to generate an RSA modulus in a distributed fashion. Alas, it might not be possible to use this method, since both [9] and [27] require that the common RSA modulus N be a product of two *safe* primes. (Informally, a large prime p is *safe* if $p = 2q + 1$ where q is itself a large prime.) Furthermore, we believe that using any method to generate RSA keys in a distributed manner involves prohibitively high communication and/or computation overhead which severely impacts the practicality of such techniques in many group setting (such as MANETs). In the next section, we present an alternative solution, based on threshold DSA, which addresses both aforementioned problems.

7 Threshold DSA (TS-DSA)

In this section, we present a scheme which uses threshold DSA signatures, denoted by TS-DSA, to implement membership control mechanism. This is an extension of the threshold DSA signature scheme presented in [10] where n, the number of group members, can be increased without changing the group secret. Similar to [10], this scheme is secure only if there are less than $t' = \lfloor \frac{t+1}{2} \rfloor$ malicious (subverted) members. Below, we mention the details of our scheme, without any formal proof of security. However, since our scheme is a simple extension of [10], we believe our scheme provides the same security as the underlying threshold DSA signature scheme.

7.1 Setup

The TS-DSA scheme can be initialized by either: 1) a trusted dealer similar to the TS-RSA scheme described above, or 2) a group of t or more founding members. Next, we consider both techniques in detail.

Initialization by dealer: TD generates DSA keys (p, q, g, x, y) where x is the group secret key. Then, TD selects a random polynomial $f(z)$ over $\mathbb{Z}_q$ of degree $t' - 1$, such that the shared secret is $f(0) = x$. In order to enable VSS, TD computes and published the witnesses $w_i = g^{f_i}$ for $(i = 0, \cdots, t' - 1)$. TD also computes the secret shares $ss_i = x_i = f(i) \pmod{q}$, and issues GMC_i for each M_i.

Self-initialization by founding members: The initial group founding members $M_i(|i| \geq t)$ select individual polynomials $f_i(z)$ over $\mathbb{Z}_q$ of degree $t' - 1$, such that $f_{i0} = x_i$. Then, using JSS, each M_i computes her own secret share ss_i, such that $ss_i = \sum_{j=1}^{l} f_j(i) \pmod{q}$ $(|l| \geq t)$. Once M_i gets her share, it is rather easy to recover the secret using Lagrange interpolation. Also, the dealing process supports VSS since each $f_i(z)$ is in $\mathbb{Z}_q$ and $g^q = 1 \pmod{p}$ (g is an element of order q in $\mathbb{Z}_p^*$). For more details, refer to [23].

7.2 GMC Issuance

Let n be the number of current group members. We assume that each $M_i(i = 1, \cdots, n)$ has the necessary group public key parameters (p, q, g) as well as her own secret share ss_i of the group secret. A coalition of t members can recover the group secret x using Lagrange interpolation. Also, unlike RSA, in order to generate a DSA signature, a *random* secret k is required. We apply JSS to *collectively* generate a random secret k. Figure 2 shows the message flow to obtain GMC using TS-DSA scheme.

$M_{new} \rightarrow M_i$:	$SIG_{new}(PKC_{new}, GID, etc.)$	(1)
$M_{new} \leftarrow M_j$:	$SIG_j(GMC_j)$	(2)
$M_{new} \rightarrow M_j$:	SL_{new}	(3)
$M_{new} \leftarrow M_j$:	$u_j,\ v_j$	(4)
$M_{new} \rightarrow M_j$:	r, GMC_REQ_{new}	(5)
$M_{new} \leftarrow M_j$:	$s_j\ \ (= k_j(m + x_j r) \mod p)$	(6)

Figure 2. TS-DSA: Certificate Issue

1. M_{new} broadcasts signed JOIN_REQ containing the public key certificate PKC_{new}, the group name, etc, to at least t current group members $M_i (|i| \geq t)$.
2. The group members, who wish to participate in the admission, reply with their respective membership certificates GMC_i to M_{new} after verifying PKC_{new} (Refer to footnote 2).
3. M_{new} picks at random t responders $M_j (j \in_R i, |j| = t)$, collects their ID_j-s from their respective GMC-s to form a signer list SL_{new} and sends it to each M_j.
4. Each M_j randomly chooses her polynomial $k_j(z),\ a_j(z)$ in $\mathbb{Z}_q$ of degree $t' - 1$. M_j computes $k_j(i)$ and $a_j(i)$ for all signers $M_i (i = 1, \cdots, t)$ in SL_{new}, and then distributes $k_j(i)$ and $a_j(i)$ to all her co-signers. After receiving her partial shares from the other co-signers, M_j computes k_j and a_j such that $k_j = k(j) = \sum_{l=1}^{t} k_l(j),\ a_j = a(j) = \sum_{l=1}^{t} a_l(j) \pmod q$. Then, M_j computes u_j and v_j such that $u_j = k_j \cdot a_j \pmod q$, $v_j = g^{a_j} \pmod p$, and sends u_j and v_j back to M_{new}. Why each M_j must take (at least) two polynomials is referred to [10].
5. M_{new} computes r without knowing k and k^{-1} as follows: First, she computes u and v such that $u = \sum_{j=1}^{t} u_j\, l_j(0) \pmod q$ which finally equals to $ka \pmod q$, $v = \prod_{j=1}^{t} (v_j)^{l_j(0)} \pmod p$ which equals $g^a \pmod p$. Next, she computes the inverse $u^{-1} \pmod q$ and finally computes r as $r = (v^{u^{-1}} \mod p) \mod q$ which equals $(g^{k^{-1}} \mod p) \mod q$. Then, M_{new} sends r to M_j, along with GMC_REQ_{new}.
6. M_j computes partial signature s_j such that $s_j = k_j(m + x_j r) \mod q$, where x_j is M_j's share of group secret x, and m is information derived from the GMC_REQ_{new}. Then, M_j sends s_j to M_{new}.
7. M_{new} computes the complete signature s such that $s = \sum_{j=1}^{t} s_j \cdot l_j(0) \pmod q$ which equals $k(m+xr) \pmod q$. That is, M_{new} obtains her own GMC_{new}.

7.3 Share Acquisition

The share acquisition procedure – through which M_{new} obtains her own new share ss_{new} from current group members – is performed as part of message (6) of certificate issue (refer to section 7.2).

t' members (M_j-s) compute partial secret share $pss_j(new)$ for M_{new} as $pss_j(new) = x_j \cdot l_j(new) \mod q$, where $l_j(new) = \prod_{l=1, l \neq j}^{t'} \frac{new - l}{j - l} \pmod q$. Each M_j sends $pss_j(new)$ to M_{new}. Then, M_{new} computes her own share ss_j by summing up $pss_j(new)$ $(j = 1, \cdots, t')$ and verifying her correctness by checking $g^{ss_{new}} = [\prod_{i=0}^{t'-1} (w_i)^{new^i}] \pmod q$.

8 Discussion

The solution presented above provides basic secret sharing functions. However it does not address proactive secret sharing as well as the issue of dynamically changing the threshold in proportion to the current group size. Although, due to space limitations, we do not elaborate on the details of these mechanisms, we note that it is possible to achieve proactivity by following the methods outlined in [18] in a fully distributed fashion. Furthermore, there have been a few results that allow changing the threshold dynamically [14, 19, 5]. In particular, we can apply the scheme proposed by Jarecki [14] which suggests reducing the degree of the secret-sharing polynomial in order to decrease the threshold and vice-versa. This causes the secret shares of all the members to be re-randomized, while the group secret remains the same[4]. However, implementing this scheme in a fully distributed manner without involving all members remains an open challenge.

9 Performance

We implemented our membership control mechanisms on Gnut-0.4.21 [11] (an open-source Gnutella [28] implementation) in order to measure their performance in the context of a *real* P2P application. We refer to the resulting software as *Secure Gnut*. At the setup phase of the Gnutella protocol, the connection is established by communicating so-called ping and pong messages which are based on IP addresses. Since the same user can submit multiple votes with different IP addresses [6], we modified our implementation so that the connection is made only if the responder answers with her valid GMC. For this purpose, we specified new messages, called sping and spong. The sping

[4] We believe that it is reasonable to assume that all adjustments to the threshold value t proceed in a lock-step fashion. In particular, each change in t causes a simultaneous change in the maximum number of members that can be compromised (or malicious) in that time-period. We emphasize that the transition from a (t_1, n_1) system to a (t_2, n_2) system is a discrete event taking place at some time T_{12}. At all times preceding T_{12}, at most $(t_1 - 1)$ compromised (or malicious) members can be present.

contains the requester's PKC, and the `spong` is composed of the responder's GMC and her signature to prove the possession of her own private key.

The cryptographic functions are developed using the OpenSSL library [22]. Our membership control toolkit is written in C on Linux, and currently consist of about 30,000 lines of code. The source codes for both the membership control toolkit and *Secure Gnut* are available in [24].

We performed measurements on the following Linux machines connected on a high-speed LAN: P4-1.2GHz, P3-977MHz, P3-933MHz, and P3-797MHz. For the sake of fairness, each machine ran the same number of Secure Gnut processes. We measured the basic operations and then compared three cryptographic protocols with both static and dynamic thresholds. We used 1024-bit modulo N in RSA and TS-RSA, and 160-bit q and 512-bit p in DSA and TS-DSA for fair comparison. Since each protocol has different number of communication rounds, we measured total processing time from sending of the `JOIN_REQ` to combining new GMCs[5].

9.1 Basic operations

Table 1 shows the time for signature generation and signature verification in terms of key size with plain RSA and TS-RSA, respectively. For both TS-RSA and TS-DSA, we set t=3. In plain RSA, the cost of signature verification is much cheaper than that of signature generation, while, in TS-RSA, verification cost is extremely expensive. This is because the computation of $m^N \pmod N$ in *t-bounded offsetting algorithm* [17] has to be done almost every time we verify the signature. Unlike private key encryption in plain RSA, we can not apply the *Chinese Remainder Theorem* to speed up computation, since the verifier does not know the prime factors of N. From our experiments, we found that this computation takes more than 95% of the total verification cost. This is a critical observation. Contrary to our intuition, the performance result of join cost shows that TS-RSA is no better than TS-DSA.

Table 1. Comparison of RSA and TS-RSA (P3-977MHz, t=3)

Key	RSA (msecs)		TS-RSA (msecs)	
(bits)	Sign	Verify	P-Sign	Verify
1024	7.395	0.249	24.093	24.802
2048	43.169	0.576	143.534	148.034
4096	278.141	1.693	971.893	981.270

Table 2 compares the performance of the standard DSA and TS-DSA in terms of the size of parameter p. Most computations in DSA are performed using a 160-bit prime q. In standard DSA measurements, signature generation is faster than signature verification. However, in TS-DSA, the costs are reversed (generation is much slower than verification). This is because JSS requires some extra exponentiations in the process of computing r in step (5) of section 7.2.

Table 2. Comparison of DSA and TS-DSA (P3-977MHz, t=3)

Key	DSA (msecs)		TS-DSA (msecs)	
(bits)	Sign	Verify	P-Sign	Verify
512	1.609	4.084	6.863	4.480
768	2.769	6.840	11.495	7.182
1024	4.017	8.494	16.880	8.870

9.2 Case I: Fixed Threshold

Figure 3 shows the cost of a new member joining the group when the group threshold is static. We measured the join cost by changing the threshold. As expected, plain RSA is the best performer in terms of computation time. However, we also see that both TS-RSA and TS-DSA exhibit reasonable costs (e.g., < 1 sec.) at least until t=20, which is large enough for membership control.

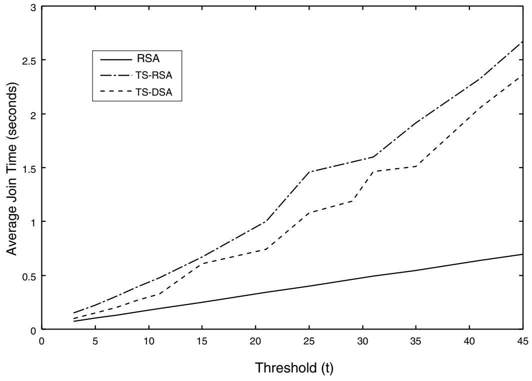

Figure 3. Join cost with fixed threshold

9.3 Case II: Dynamic Threshold

In most P2P systems, such as Gnutella, group size can fluctuate drastically within a short time. As the group grows or shrinks, we need to increase or decrease the threshold without changing the current group secret. Changing the threshold dynamically requires all current members to update their shares. Any t members can collaboratively increase or decrease the current threshold without keeping track of the previous polynomial. This is a relatively simple process. However, in Gnutella, it poses a problem. The reason being that, after updating the polynomial collectively, the t members must broadcast the update to other $n - t$ members in a *secure* and *synchronous* fashion. While this may be feasible in peer group settings which have an underlying reliable, synchronous group communication system, there is no protocol message in Gnutella to achieve this goal. Therefore, in our simulation, we only measure the cost of actually updating the threshold, without considering the associated communication costs.

Additionally, since updating the threshold is an expensive operation, it is not practical if every membership event

[5] In these experiments we did not consider the *partial share shuffling* for both TS-RSA and TS-DSA.

triggered an update process. In order to prevent this, we apply a simple *window* mechanism. Specifically, every member keeps state of n_{old}, which is the group size at the time of the last threshold-update process. A new threshold-update process is triggered only when the difference between the current group size n_{cur} and n_{old} is greater then Win where Win is the window buffer. In other words, threshold update process is triggered only when $|n_{cur} - n_{old}| \geq Win$.

In Figure 4, we show the cost of a new member joining the group when the group threshold varies dynamically according to current group size. In this experiment, the participating ratio (R) of the threshold is set to 20% of the current group size. Note the ratio is dependent on admission policy. The threshold is determined by multiplying the group size by R. We measured the performance until n=50. At n=50, threshold is set to 10, the cost is less than 0.5 sec with all mechanism. We also figure out the extra cost is added whenever threshold changes in the figure.

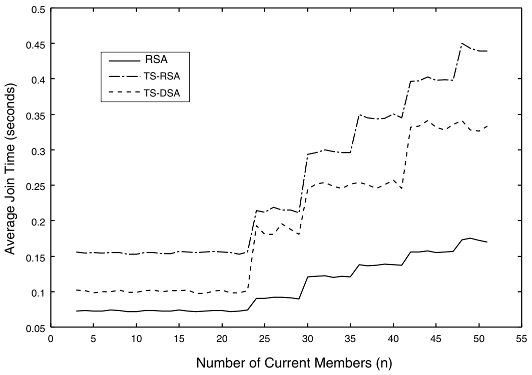

Figure 4. Join cost with dynamic threshold ($R = 20\%$)

10 Conclusion

In this paper, we investigated the usefulness of threshold cryptography in providing secure membership control for dynamic peer groups common in P2P systems and MANETs. Specifically, we examined two threshold signature schemes: one based on RSA signatures and the other based on DSA signatures and measured their performance. We discovered that the threshold RSA scheme contains a security flaw and, hence, cannot be used in a group setting with potential malicious insiders. The threshold DSA scheme overcomes this problem.

Perhaps more importantly, our work can be viewed as a *negative result* as it casts doubt upon the practicality of using threshold cryptography in peer-to-peer and MANET settings. Nevertheless, further exploration of new, more efficient, threshold techniques is needed as well as further experimentation with, and careful assessment of, existing methods.

References

[1] D. Boneh and M. Franklin. Efficient generation of shared rsa keys. In *CRYPTO*, 1997.

[2] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *FOCS*, 1985.

[3] Y. Desmedt. Society and group oriented cryptosystems. In *CRYPTO*, 1987.

[4] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, 1989.

[5] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Tech Report TR-97-01, GMU, 1997.

[6] J. R. Douceur. The sybil attack. In *IPTPS*, 2002.

[7] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS*, 1987.

[8] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal-resilience proactive public-key cryptosystems. In *FOCS*, 1997.

[9] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of rsa functions. In *CRYPTO*, 1996.

[10] R. Gennaro, S.Jarecki, H.Krawczyk, and T.Rabin. Robust threshold DSS signatures. In *EUROCRYPT*, 1996.

[11] Gnut v0.4.21, `http://schnarff.com/gnutelladev/source/gnut`.

[12] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing, or: How to cope with perpetual leakage. In *CRYPTO*, 1995.

[13] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 public key infrastructure certificate and CRL profile. RFC 3280, 2002.

[14] S. Jarecki. Efficient threshold cryptosystems, MIT PhD thesis, 2001.

[15] Y. Kim, D. Mazzocchi, and G. Tsudik. Admission control in peer groups. In *IEEE NCA*, 2003.

[16] J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, and S. Lu. Adaptive security for multi-level ad-hoc networks. In *Journal of Wireless Communications and Mobile Computing*, volume 2, 2002.

[17] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *IEEE ICNP*, 2001.

[18] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self-securing ad hoc wireless networks. In *ISCC*, 2002.

[19] K. Martin, R. Safavi-Naini, and H. Wang. Bounds and techniques for efficient redistribution of secret shaers to new access structures. *The Computer Journal*, 1999.

[20] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. 1997. ISBN 0-8493-8523-7.

[21] NIST. Digital Signature Standard. Technical Report 169, 1991.

[22] OpenSSL Project, `http://www.openssl.org/`.

[23] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT*, 1991.

[24] Peer Group Admission Control Project, `http://sconce.ics.uci.edu/gac`.

[25] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2), 1978.

[26] A. Shamir. How to share a secret. *Commun. ACM*, 22(11), 1979.

[27] V. Shoup. Practical threshold signatures. In *EUROCRYPT*, 2000.

[28] The Gnutella Protocol Specification v0.4, `http://www.clip2.com/GnutellaProtocol04.pdf`.

[29] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6), 1999.