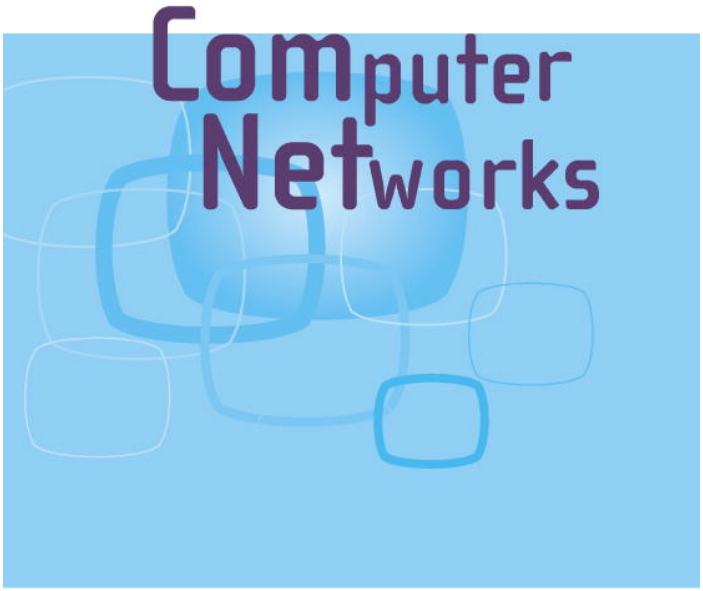
Volume 51 Issue 12 22 August 2007 ISSN 1389-1286



Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 3632–3649
Computer Networks
www.elsevier.com/locate/comnet

Threshold cryptography in P2P and MANETs: The case of access control ☆

Nitesh Saxena [a], Gene Tsudik [b], Jeong Hyun Yi [c,*,1]

[a] Computer and Information Science, Polytechnic University, United States
[b] Computer Science Department, University of California, Irvine, United States
[c] Communication and Networking Lab, Samsung Advanced Institute of Technology, Republic of Korea

Received 4 September 2006; received in revised form 29 January 2007; accepted 9 March 2007
Available online 23 March 2007

Responsible Editor: L. Salgarelli

Abstract

Ad hoc groups, such as peer-to-peer (P2P) systems and mobile ad hoc networks (MANETs) represent recent technological advancements. They support low-cost, scalable and fault-tolerant computing and communication. Since such groups do not require any pre-deployed infrastructure or any trusted centralized authority they have many valuable applications in military and commercial as well as in emergency and rescue operations. However, due to lack of centralized control, ad hoc groups are inherently insecure and vulnerable to attacks from both within and outside the group.

Decentralized *access control* is the fundamental security service for ad hoc groups. It is needed not only to prevent unauthorized nodes from becoming members but also to bootstrap other security services such as key management and secure routing. In this paper, we construct several distributed access control mechanisms for ad hoc groups. We investigate, in particular, the applicability and the utility of threshold cryptography (more specifically, various flavors of existing threshold signatures) towards this goal.

Keywords: Mobile ad hoc networks; Key management; Access control; Threshold cryptography; Membership management

1. Introduction

Ad hoc groups, such as peer-to-peer (P2P) systems and mobile ad hoc networks (MANETs), are very popular in today's computing, especially in the research community. They lack infrastructure and do not need any trusted authority. Moreover, they are inherently scalable and fault tolerant. Such characteristics find many interesting applications in military and commercial settings as well as in emergency and rescue operations. However, their open nature and lack of centralized control result in some security challenges.

The security research community recognized the need for specialized security services in ad hoc

☆ Portions of this paper appeared in [26,37,38].

* Corresponding author. Tel.: +82 10 2481 0826; fax: +82 31 280 9555.
E-mail addresses: nsaxena@duke.poly.edu (N. Saxena), gts@ics.uci.edu (G. Tsudik), jeong.yi@samsung.com (J.H. Yi).

[1] This work has been done while at UC Irvine.

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.03.001

groups. *Access Control* is particularly important since most other traditional security are based upon it. In this context, an access control mechanism must prevent unauthorized nodes from becoming a part of the group and to establish trust among members in the absence of a trusted authority. Access control is also essential to bootstrap other security services, such as secure group communication [44,43] and secure routing (in MANETs), such as Ariadne [17], SPINS [33], etc.

The concept of threshold cryptography involves distributing cryptographic primitives (such as decryption and digital signatures) in order to secure them against a corruption of a certain number of parties, i.e., a threshold. For example, a $(t-1,n)$ threshold signature scheme [6] allows, in a group of a total of n parties, to share the ability to digitally sign messages in such a way that any t parties can do so jointly, whereas no coalition of up to $t-1$ parties can.

1.1. Related work

Zhou and Haas [45] first suggested using threshold cryptography [45] to secure mobile ad hoc networks. Their intuition was to distribute trust among the nodes of the network such that no less than a certain threshold of nodes are trusted. They proposed a distributed certification authority (CA) [16] which issues certificates (using some threshold signature [6] protocol) to the nodes joining the network. Certificates enable the nodes to communicate with each other in a secure and authenticated manner. This work also led to the development of COCA [46], an online certification authority for wired networks. Although attractive, this idea is not applicable to ad hoc groups. Their approach is hierarchical: only select nodes can serve as part of the certification authority and thus take part in admission decisions. Moreover, contacting the distributed CA nodes in a MANET setting is difficult since such nodes might be many hops away.

Luo et al. considered the same problem in [23] and Kong et al. in [21,20] as well as [24,22]. This body of work proposed a set of protocols for ubiquitous and robust access control in MANETs. They amended the model of Zhou and Haas to allow every member to participate in access control decisions, thus maintaining the true "peer" nature of ad hoc groups and providing increased availability. Unfortunately, this otherwise elegant scheme has been shown to be insecure [26,18].

Recently, Kim et al. [19] developed a group access control framework based on a menu of cryptographic techniques. This framework classifies group admission policy according to the entity (or entities) making admission decisions. The classification included simple access control policies, such as static ACL (Access Control List)- or attribute-based admission, as well as admission based on the decision of a fixed entity: external (e.g., a CA or a TTP) or internal (e.g., a group founder). Such simple policies are relatively easy to support and do not present much of a technical challenge. However, they are inflexible and ultimately unsuitable for dynamic ad hoc networks. Static ACLs enumerate all possible members and hence cannot support truly dynamic membership (although they work well for closed networks). Admission decisions made by a TTP or a group founder violate the peer nature of the underlying ad hoc group.

In our prior work [37], we constructed access control mechanisms based on plain RSA signatures and accountable subgroup multisignatures [28]. However, we realize that such mechanisms have *lineage* problem. This problem occurs when a membership certificate is issued to a new member: each member (sponsor) who takes part in the admission process needs to confirm (by signing) its agreement to admit this new member. Essentially, a membership certificate has to be signed by some number of membership sponsors.[2] However, each sponsor needs to attach its own certificate to its signature on a new member's certificate in order to make group certificates universally verifiable. However, a sponsor's own certificate also has to be countersigned by its erstwhile sponsors, and so on, and so forth. This is clearly unworkable since a member's certificate would have to be accompanied by a number of certificate chains that affirm its lineage.

1.2. Our contributions

In this paper, we explore the utility of threshold cryptography (more specifically various existing threshold signatures) in constructing decentralized access control mechanisms for ad hoc groups. We first point out the inapplicability of known threshold RSA signatures towards this goal, and carry

[2] This number is determined by the group admission policy; common examples are a certain fraction of current members or a fixed threshold. See [19] for a detailed discussion of admission policies.

on to build access control mechanisms based on various flavors of discrete logarithm based threshold signatures, namely, threshold DSA [11], threshold Schnorr [12], and threshold BLS [2]. We compare and evaluate these mechanisms, via theoretical and experimental analysis in a real MANET setting.[3]

1.3. Scope

Group access control is a broad topic which includes access control mechanisms and the more general issue of group security policy. This work is concerned only with access control 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 ad hoc group. Moreover, although we recognize that proactivity [30,15,14] is an important issue to cope up with stronger mobile adversaries, we do not consider it here. However, the threshold signature schemes that we employ have proactive support and the software implementation of the same is left as an avenue for future work.

Our work uses group membership certificates to assert group membership. Certificates, as usual, prompt the revocation headache. However, certificate revocation issues are beyond the scope of this work.

1.4. Organization

The rest of the paper is organized as follows: Section 2 summarizes notations. Section 3 provides the high-level description of the access control protocol. Next, Section 4 discusses inapplicability of known threshold RSA schemes and points out the robustness problem with a recently proposed threshold RSA signature scheme [21]. Sections 5–7 present the proposed access control protocols based on different threshold signature schemes. Then, experimental results are presented and analyzed in Section 8. (Appendix A recalls some cryptographic primitives and Appendix B summarizes the threshold RSA scheme of [21]).

[3] Although in this paper we report only on the experimental evaluation performed in a MANET setting, our access control mechanisms are also generally applicable in various P2P systems.

Table 1
Notation

P_i	Group member i
id_i	Identity for P_i
t	Admission threshold
n	Total number of group members
$\mathbb{G}$	Cyclic group in finite fields
g	Generator of group $\mathbb{G}$
$\mathbb{G}_1, \mathbb{G}_2$	Cyclic GDH groups of order q
P	Generator of group $\mathbb{G}_1$
$\hat{e}$	Bilinear map s.t. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$
H	Hash function such as SHA-1 or MD5
H_1	Special hash function s.t. $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$
x_i	Secret share of P_i
$x_i^{(j)}$	Partial share for P_i by P_j
GMC_i	Group membership certificate for P_i
SL_i	List of sponsors for P_i
$S_i(m)$	P_i's signature on message m
K_{ij}	Pairwise key between P_i and P_j
$E_{K_{ij}}$	Encryption with K_{ij}
$\phi(N)$	Euler's Totient for RSA modulus N

2. Notation

Notation used in this paper is summarized in Table 1. In the rest of the paper, we use the terms member/node/player and group/network/system interchangeably.

3. Group access control

A threshold signature scheme enables any subgroup of t members in a group to collaboratively sign messages on behalf of that group. This is achieved by secret-sharing the signature key among the group members, and allowing them to compute a signature on some message via a distributed protocol in which the members use the shares of the signature key instead of the key itself. Threshold signature schemes can tolerate up to $t-1$ corruptions in the whole lifetime of the system.

The idea of threshold signatures applies directly to build access control mechanisms by making collaborative decisions. Next, we overview a generic access control protocol. Similar to the security model of underlying threshold signature schemes, in our access control mechanisms we consider an adversary who is capable of corrupting at most $t-1$ members and tries to come up with an existential forgery under the adaptively chosen message attack model.

The access control mechanism is initiated by a prospective member or an "applicant". At the

end, given that enough honest members ($\geqslant t$) approve admission, the applicant becomes a member of the group and possesses its share of the group secret (called "secret share") and its group membership certificate (GMC).

1. *Bootstrapping:* The group is initialized by either a trusted dealer or a set of founding members. The dealer or founding members initialize the group by choosing a group secret key, and computing and publishing the corresponding public parameters in the group certificate. The group secret is shared among the founding member(s) using either Shamir's threshold secret sharing (TSS) or joint secret sharing (JSS) techniques presented in Appendix A.1 or Appendix A.2, respectively.
2. *Member admission:* A prospective member P_{n+1} who wishes to join the group must be issued its secret share and membership certificate by current members. Fig. 1 gives a high-level view of group admission protocol.
   - P_{n+1} initiates the admission protocol by sending a `JOIN_REQ` message to the group.
   - A member, that receives this `JOIN_REQ` message and approves the admission of P_{n+1}, replies, over a secure channel, with a partial secret share and a partial signature derived from its secret share for P_{n+1}.
   - Once P_{n+1} receives partial shares and signatures from at least t different members, it uses them to compute its secret share and membership certificate.
   - Finally, P_{n+1} verifies the validity of its reconstructed secret share and group membership certificate before using them. Also, when P_{n+1} detects that its secret share or membership certificate is invalid, it must be able to identify the bogus partial share(s) and/or partial signature(s), and thus trace the malicious group member(s).

   Note that, this step may involve multiple rounds and/or co-ordination among the members who commit to the requesting member, depending on the underlying cryptographic techniques.
3. *Membership authentication:* To ensure only genuine members are involved in communication, every member must be able to prove membership to other members.

4. Threshold RSA schemes

Various flavors of threshold signatures exist in literature: RSA based, DSA based, Schnorr based and more recently, BLS [4] based. However, known provably secure threshold RSA signatures *do not* yield access control mechanisms for ad hoc groups. In this section, we begin by carefully considering various threshold RSA schemes, explain why they are not applicable for access control in ad hoc groups, and point out the robustness problem with a recently proposed threshold RSA scheme [21,20,24,22].

4.1. Analysis of known schemes

Several threshold RSA signatures are proposed in literature [9,10,35,42,21,20,24,22] that might be used to construct the group access control protocol. Unfortunately, *none* of these schemes are directly applicable.

1. *Schemes by Frankel et al. and Rabin.* The currently known provably secure threshold RSA signature schemes, two schemes by Frankel et al. [9,10] and a scheme by Rabin [35], are not applicable for access control in ad hoc groups.

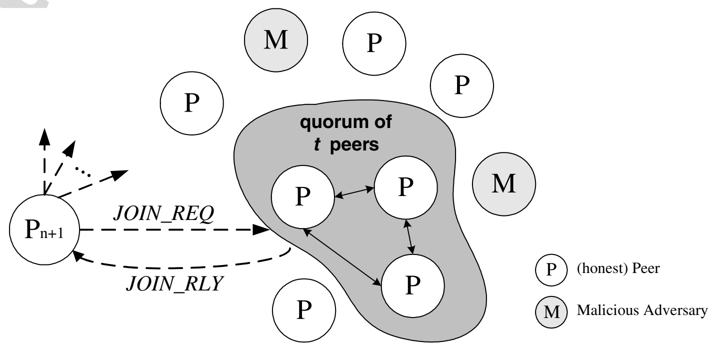


Fig. 1. Group admission protocol.

In particular, the RSA signature scheme of [9] is practical only for small groups, while in the other two provably secure threshold RSA schemes known today [10,35] (which employ *additive secret sharing* as opposed to polynomial secret sharing of [41]) the members participating in the threshold signature protocol need to reconstruct the secret shares of the group members that are currently inaccessible to them. In this way both protocols essentially equate a temporarily inaccessible group member with a corrupt one, whose secrets might just as well be fabricated. This is an undesirable feature for asynchronous ad hoc groups where members are often inaccessible to one another. In such settings we need to enable isolated but large enough subgroups of members to operate without reconstructing everyone else's secrets.

2. *Scheme by Shoup.* Another well known and more recent provably secure threshold RSA scheme was proposed by Shoup [42]. This scheme is more elegant than the above ones because the signature generation and verification is fully non-interactive and it also avoids the inaccessibility problem by employing the polynomial (t,n) secret sharing of Shamir [41]. However, since the secret sharing is performed over secret modulo $\phi(N)$ (unlike over publicly known integers in the schemes discussed above), it is not possible for the group members to provide a new member with its secret share. Moreover, Shoup's scheme requires a trusted dealer to generate the RSA keys, which is an undesirable feature in ad hoc groups. Boneh and Franklin [3] developed a method to generate an RSA modulus in a distributed fashion. Alas, it might not be possible to use this method, since Shoup's scheme requires that the common RSA modulus N be a product of two *safe* primes.[4] 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.
3. *Scheme by Kong et al.* In an effort to mitigate the above problem of the known threshold RSA signatures, Kong et al. [21] proposed a new threshold RSA scheme, geared toward providing security services in MANETs. Unfortunately, this scheme, contrary to what its authors claimed, is neither robust (i.e., it cannot tolerate malicious group members) nor secure. We first pointed out the robustness problem in [26]. This problem is presented in detail in the following section. We also presented an attack on the scheme in which an admissible threshold of malicious group members can completely recover the RSA secret key in the course of the lifetime of this scheme [18].

[4] Informally, a large prime p is *safe* if $p = 2q + 1$ where q is itself a large prime.

4.2. Robustness problem with URSA RSA signature scheme

Recently, Kong et al. in a series of papers [21,20,24,22] proposed a set of protocols for providing ubiquitous and robust access control, so-called *URSA*, in MANETs without relying on a centralized authority. In this section we argue that this scheme is not robust against malicious adversaries [26], i.e., it *fails to provide* the verifiability of partial secret shares $(x_{n+1}^{(j)} - s)$ as well as combined secret share (x_{n+1}) and the partial signatures $(s_j - s)$. As a result, malicious or compromised members can send fake shares and fake signatures to new members without being detected and in turn disrupt the admission service. The reason is as follows:

1. *Verifiability of partial secret shares and combined secret share.*
   Since $x_{n+1}^{(j)}$s and x_{n+1} are computed modulo N and not $\phi(N)$, it is impossible to verify the correctness using publicly known witnesses. 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, we cannot apply verifiability mechanisms VPSS and VSS (over modulus N) described in Appendixes A.7 and A.4 to determine the correctness of the partial secret shares and the secret share respectively. In other words,

$$g^{x_{n+1}^{(j)}} \neq \left[\prod_{k=0}^{t-1} (W_k)^{id_{n+1}^k}\right]^{\lambda_j(id_{n+1})} \pmod N \tag{1}$$

   and

$$g^{x_{n+1}} \neq \prod_{k=0}^{t-1} (W_i)^{id_{n+1}^k} \pmod N. \tag{2}$$

   Example. 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 P_7 receives the following partial shares from t existing members P_2, P_3, and P_5: $x_7^{(2)} = 71$, $x_7^{(3)} = 74$, and $x_7^{(5)} = 72 \pmod{119}$. P_7 computes its share $x_7 = x_7^{(2)} + x_7^{(3)} + x_7^{(5)} = 98 \pmod{119}$. To check verifiability of the secret share, he computes $g^{x_7} = 3^{98} = 9 \pmod{119}$.
Also, using the witness values, P_7 can get the right hand side of Eq. (2) as follows:

$$(W_0)(W_1)^7(W_2)^{7^2} = 1 \pmod{119}.$$

Therefore, even though x_7 is correctly computed, $g^{x_7} \neq \prod_{i=0}^{2}(W_i)^{7^i} \pmod{119}$.

2. *Verifiability of partial signatures.* In case the signature reconstruction fails, P_{n+1} must verify the correctness of each partial signature s_j and trace the faulty signer(s) in the process. This involves a zero knowledge proof of knowledge (ZKPK) protocol [40] $\{ZKPK(d_j : s_j = m^{d_j} \pmod N \wedge W_j = g^{d_j} \pmod N))\}$ between P_{n+1} and the signer P_j. However, due to the reasons explained above, it is impossible to compute W_j using the public witness values. Explicitly,

$$W_j \neq \left[\prod_{k=0}^{t-1}(W_k)^{id_j^k}\right]^{\lambda_j(0)} \pmod N \qquad (3)$$

and therefore, performing the above ZKPK protocol is meaningless. Thus, it is impossible for the prospective member to trace the faulty signer(s).

5. Threshold DSA based access control

In this section, we describe the access control mechanism (referred to as TS-DSA) [26,37] based on the threshold DSA scheme [11] of Gennaro et al.

5.1. DSA

The DSA [27] is a signature scheme based on the El Gamal signature scheme [7]. In our description of the TS-DSA protocol we follow the notation introduced in [11], which differs from the original presentation by switching k and k^{-1}. This change allows a clearer presentation.

1. *Key Generation.* DSA uses the system-wide parameters (p, q, g) where q is a 160-bit prime number, p is a large prime number such that q divides $(p - 1)$, and g is an element of order q in $\mathbb{Z}_p^*$. Each user selects a random integer $x \in \mathbb{Z}_q$ as a private key and computes a public key y such that $y = g^x \pmod p$.
2. *Signing.* To generate a signature on the message m, the signer picks a random number $k \in \mathbb{Z}_q$ and calculates $r = (g^{k^{-1}} \pmod p) \bmod q$ and $s = k(m + xr) \pmod q$. The signature for m is the pair (r, s).
3. *Verification.* To verify a DSA signature, the verifier computes $r' = (g^{ms^{-1}} y^{rs^{-1}} \pmod p) \bmod q$ and checks if $r' = r$.

5.2. Bootstrapping

TS-DSA can be initialized by either: (1) a trusted dealer or (2) a group of $3t - 2$ or more founding members.[5]

1. *Centralized Initialization.* The trusted dealer *TD* does the following:
   (a) The *TD* generates the system parameters (p, q, g), selects a random polynomial $f(z) = a_0 + a_1 z + \cdots + a_{t-1} z^{t-1}$ over $\mathbb{Z}_q$ of degree $(t - 1)$ such that $f(0) = a_0 = x$ where a_is, for $i \in [0, t - 1]$, are the coefficients of the polynomial and x is a group secret. In order to enable VSS (refer to Appendix A.4) the *TD* computes and publishes the witnesses $W_k = g^{a_k}$ for $k \in [0, t - 1]$. Note that the witness value $W_0 = g^x$, also denoted by y, is actually the group public key.
   (b) For each P_i $(i \in [1, 3t - 2])$, the *TD* computes the secret share x_i such that $x_i = f(id_i) \pmod q$ and issues the group membership certificate GMC_i. Note that the *TD* is not required hereafter.
2. *Decentralized Initialization.* A set of founding members P_i $(i \in [1, 3t - 2])$ do the following:
   (a) Each P_i selects an individual polynomial $f_i(z)$ over $\mathbb{Z}_q$ of degree $(t - 1)$ as in JSS protocol (refer to Appendix A.2).

[5] Since threshold schemes must tolerate up to $t - 1$ corruptions, it is assumed that there are at most $(t - 1)$ malicious or faulty members. Hence, the minimum number of founding members is $3t - 2$; i.e., $(2t - 1)$ non-faulty members (as required in threshold DSA [11]) and $(t - 1)$ faulty members.

(b) Then, each P_i computes its own secret share x_i such that $x_i = \sum_{j=1}^{3t-2} f_j(id_i) \bmod q$. Note that during this process the VSS protocol is applied to check the validity of x_i.
(c) Now, in order to provide each member with a membership certificate, any set of $(2t-1)$ founding members must collaborate.

5.3. Member admission

Let n ($\geqslant 3t-2$) be the number of current group members. The prospective member P_{n+1} invokes the admission process described in Section 3. The detailed steps of the protocol are described below. Fig. 2 shows the message flow of the TS-DSA protocol required to obtain GMC_{n+1}.

1. A prospective member P_{n+1} broadcasts signed JOIN_REQ[6] message m containing its identity certificate PKC_{n+1} which contains its public key PK_{n+1} and identity id_{n+1} in order to prove the knowledge of the corresponding private key.[7]
2. After verifying the signed JOIN_REQ, group members[8] who wish to participate in the admission of P_{n+1} reply with a signed message[9] containing their respective membership certificates GMC_is which include id_i where $i \in [1, t']$ and $2t-1 \leqslant t' \leqslant n$.
3. P_{n+1} picks at random $(2t-1)$ out of t' ($\geqslant 2t-1$) sponsors, P_js, collects their id_js from their respective GMC_js to form a sponsor list SL_{n+1} and replies with a signed acknowledgment message to each of them.
4. Each P_j randomly chooses polynomials $k_j(z)$ and $b_j(z)$ in $\mathbb{Z}_q$ of degree $(t-1)$. P_j computes $k_j(id_i)$ and $b_j(id_i)$ for all co-signers P_i ($i \in [1, 2t-1]$) in SL_{n+1}, and then distributes $k_j(id_i)$ and $b_j(id_i)$ to them using JSS protocol. Also, the witness values for these polynomials are broadcast by each P_j in order to enable VSS. After receiving the partial shares from other co-signers, P_j computes its shares: $k_j = k(id_j) = \sum_{l=1}^{2t-1} k_l(id_j) \pmod q$ and $b_j = b(id_j) = \sum_{l=1}^{2t-1} b_l(id_j) \pmod q$. Then, P_j computes $u_j = k_j b_j$ and $v_j = g^{b_j}$, and sends them back to P_{n+1}.
5. P_{n+1} now computes: $u = \sum_{j=1}^{2t-1} u_j \lambda_j(0) \pmod q$ and $v = \prod_{j=1}^{2t-1} (v_j)^{\lambda_j(0)} \pmod p$ which finally equals to kb and g^b, respectively. Next, P_{n+1} computes the inverse $u^{-1} \pmod q$ and finally computes r such that $r = (v^{u^{-1}} \bmod p) \bmod q$ which equals $(g^{k^{-1}} \bmod p) \bmod q$. Then, P_{n+1} sends r to P_j. Note that in order to compute r without revealing any information about k or k^{-1}, each P_j must choose two polynomials [11].
6. P_j computes a partial signature $s_j = k_j(m + x_j r) \pmod q$ and sends it to P_{n+1}. P_j also sends to P_{n+1} its *shuffled* partial secret share $\tilde{x}_{n+1}^{(j)}$ for P_{n+1} using the PSRS protocol (refer to Appendix A.6): $\tilde{x}_{n+1}^{(j)} = x_j \lambda_j(id_{n+1}) + R_j \lambda_j(0) \pmod q$, where $\lambda_j(x) = \prod_{l=1, l\neq j}^{t} \frac{x - id_l}{id_j - id_l} \pmod q$ and R_j is a random share of the shared secret zero. Note that the computation of $\lambda_j(x)$ only requires t members. Then, P_j computes a pairwise key K_{jn+1} using the technique in [5], and sends the encrypted s_j and $\tilde{x}_{n+1}^{(j)}$ to P_{n+1}.
7. P_{n+1} computes the threshold signature $s = \sum_{j=1}^{2t-1} s_j \lambda_j(0) \pmod q$ which equals $k(m + xr) \pmod q$. Also, P_{n+1} computes its own share x_{n+1} by summing up $\tilde{x}_{n+1}^{(j)}$ for $j \in [1, t]$. P_{n+1} verifies the reconstructed signature (r, s) and the reconstructed share x_{n+1}, using the standard DSA verification and the VSS protocol, respectively. If these verifications succeed, P_{n+1} creates its membership certificate GMC_{n+1} which contains $m = \{id_{n+1}, PK_{n+1}, \text{etc.}\}$ and its signature (r, s). Otherwise, the technique for tracing malicious members must be employed as follows:
   (a) *Partial share tracing.* Correctness of the partial secret share $\tilde{x}_{n+1}^{(j)}$ can be checked using the VPSS technique discussed in Appendix A.7.
   (b) *Partial signature tracing.* Correctness of each individual partial signature s_j is verified by $g^{ms_j^{-1}} \cdot y_j^{rs_j^{-1}} = g^{k_j^{-1}}$, where $y_j = g^{x_j}$ using the

[6] We note that it is necessary to include timestamps, nonces and protocol message identifiers in order to secure the protocol against *replay* attacks [25]. However, we omit these values to keep our description simple.
[7] We assume that there exists an offline CA that issues long-term certificates to each node. In practice, the existence of such a CA can be avoided – a secret channel between the joining node and other nodes can be established by making use of physical out of band channels, as in [36,13]. Of course, this way of establishing secret channels would require more overhead and some user involvement.
[8] We note that multiple sponsors may reply in parallel.
[9] Note that how to provide authenticity of protocol messages is implementation-dependent. Here, the signature can be replaced with a message authentication code (MAC) [25], provided that Diffie-Hellman key exchange is available.

Step 1 (JOIN_REQ). P_{n+1} broadcasts:

$$P_{n+1} \xrightarrow{m=\{PKC_{n+1},etc.\},\ S_{n+1}(m)} \{P_1, \cdots, P_n\}$$

Step 2 (JOIN_RLY). Each P_i ($i \in [1, t']$) where $2t-1 \leq t' \leq n$ replies with their respective membership certificates GMC_i and its signature.

$$P_{n+1} \xleftarrow{GMC_i,\ S_i(GMC_i)} \{P_1, \cdots, P_{t'}\}$$

Step 3 (CHAL_REQ). P_{n+1} picks at random $(2t-1)$ out of t' sponsors and forms a list SL_{n+1} with their id_i-s from GMC_i-s.

$$P_{n+1} \xrightarrow{SL_{n+1},\ S_{n+1}(SL_{n+1})} \{P_1, \cdots, P_{2t-1}\}$$

Step 4 (CHAL_RLY). Each P_j ($j \in [1, 2t-1]$):

- computes shares k_j and b_j using JSS protocol,
- computes $u_j = k_j b_j \pmod q$ and $v_j = g^{b_j} \pmod p$.

$$P_{n+1} \xleftarrow{u_j,\ v_j,\ S_j(u_j,v_j)} \{P_1, \cdots, P_{2t-1}\}$$

Step 5 (SIGN_REQ). P_{n+1}:

- computes $u = \sum_{j=1}^{2t-1} u_j \lambda_j(0) \pmod q \ \ [= kb]$,
- computes $v = \prod_{j=1}^{2t-1} (v_j)^{\lambda_j(0)} \pmod p \ \ [= g^b]$,
- computes $r = (v^{u^{-1}} \mod p) \mod q \ \ [= g^{k^{-1}}]$.

$$P_{n+1} \xrightarrow{r,\ S_{n+1}(r)} \{P_1, \cdots, P_{2t-1}\}$$

Step 6 (SIGN_RLY). Each P_j ($i \in [1, 2t-1]$):

- computes partial signature $s_j = k_j(m + x_j r) \pmod q$,
- computes its *shuffled* partial secret share $\tilde{x}_{n+1}^{(j)}$ using PSRS protocol (described in Appendix A.6),
- encrypts s_j and $\tilde{x}_{n+1}^{(j)}$.

$$P_{n+1} \xleftarrow{E_{K_{jn+1}}(s_j,\ \tilde{x}_{n+1}^{(j)})} \{P_1, \cdots, P_{2t-1}\}$$

Step 7. P_{n+1}:

- computes the signature $s = \sum_{j=1}^{2t-1} s_j \lambda_j(0) \pmod q$,
- computes its own share $x_{n+1} = \sum_{j=1}^{t} \tilde{x}_{n+1}^{(j)} \pmod q$.

Fig. 2. TS-DSA admission protocol.

witnesses for VSS as in [5] and $g^{k_j^{-1}}$ is computed as $v_j^{u_j^{-1}}$ from the steps (2) and (5) in Fig. 2.

If either of the above tracing functions fail, P_{n+1} concludes that P_j is cheating.

5.4. Membership authentication

Every legitimate member is able to prove its membership using its own membership certificate. This involves a verifier sending a challenge to the group member, and the group member responding back with a signed challenge along with its membership certificate. The verifier first verifies the DSA signature on the certificate (using the public key of the group) and then the signature on the challenge (using the group member's public key extracted from its membership certificate).

6. Threshold Schnorr based access control

In this section, we describe the access control mechanism (referred to as TS-Sch) based on the threshold Schnorr scheme [12].

6.1. Schnorr signature scheme

The Schnorr signature scheme [39] is a variant of the El Gamal scheme and its security is based on the DL assumptions [34].

1. *Key generation.* The method to generate keys is the same as DSA key generation, except that there are no constraints on the sizes of p and q.
2. *Signing.* To generate a signature, the signer selects a random secret integer $k \in \mathbb{Z}_q$, computes $r = g^k \bmod p$, $e = H(m\|r)$, and $s = k - ex \bmod q$. The pair (e, s) is the signature of the message m.
3. *Verification.* To verify a signature, the verifier computes $r' = g^s y^e \bmod p$ and $e' = H(m\|r')$ and accepts the signature if and only if $e' = e$.

6.2. Bootstrapping

TS-Sch can be initialized in the same way as TS-DSA, except the minimum number of founding members required for decentralized initialization is $(2t - 1)$.

6.3. Member admission

Let n ($\geqslant 2t - 1$) be the number of current group members. The protocol steps are described below and in Fig. 3.

1. Same as the step (1) in Section 5.3.
2. Each P_i ($i \in [1, t']$) where $t \leqslant t' \leqslant n$, who participates in the admission of P_{n+1} randomly chooses k_i in $\mathbb{Z}_q$, computes r_i such that $r_i = g^{k_i} \pmod p$, and then replies with a signed message containing r_i and GMC_i.
3. P_{n+1} picks at random t out of t' ($\geqslant t$) sponsors, P_js and collects id_js from the respective GMC_js to form a sponsor list SL_{n+1}. Also, P_{n+1} computes $r = \prod_{j=1}^{t} r_j \pmod p$ and $e = H(m\|r)$. Then, P_{n+1} replies with a signed acknowledgment message containing e, SL_{n+1} to each of the t members.
4. Each P_j then computes the partial signature s_j and the shuffled partial secret share $\tilde{x}_{n+1}^{(j)}$ such that $s_j = k_j + ex_j \pmod q$ and $\tilde{x}_{n+1}^{(j)} = x_j\lambda_j(id_{n+1}) + R_j\lambda_j(0) \pmod q$. Then P_j sends s_j and $\tilde{x}_{n+1}^{(j)}$ to P_{n+1} over secure channel with K_{jn+1} as in the step (6) in Section 5.3.

Step 1 (JOIN_REQ). P_{n+1} broadcasts:

$$P_{n+1} \xrightarrow{m=\{PKC_{n+1}, etc.\},\ S_{n+1}(m)} \{P_1, \cdots, P_n\}$$

Step 2 (JOIN_RLY). Each P_i ($i \in [1, t']$) where $t \leq t' \leq n$ computes $r_i = g^{k_i}$ where $k_i \in_R \mathbb{Z}_q$ and then replies with their respective membership certificates GMC_i and r_i.

$$P_{n+1} \xleftarrow{r_i,\ GMC_i,\ S_i(r_i,\ GMC_i)} \{P_1, \cdots, P_{t'}\}$$

Step 3 (SIGN_REQ). P_{n+1} picks at random t out of t' sponsors and forms a list SL_{n+1} with their id_i-s from GMC_i-s, and computes $e = H(m||r)$ where $r = \prod_{j=1}^{t} r_j$

$$P_{n+1} \xrightarrow{SL_{n+1},\ e,\ S_{n+1}(SL_{n+1},\ e)} \{P_1, \cdots, P_t\}$$

Step 4 (SIGN_RLY). Each P_j ($j \in [1, t]$):

- computes partial signature $s_j = k_j + ex_j \pmod q$,
- computes its *shuffled* partial secret share $\tilde{x}_{n+1}^{(j)}$ using PSRS protocol,
- encrypts s_j and $\tilde{x}_{n+1}^{(j)}$.

$$P_{n+1} \xleftarrow{E_{K_{jn+1}}(s_j,\ \tilde{x}_{n+1}^{(j)})} \{P_1, \cdots, P_t\}$$

Step 5. P_{n+1}:

- computes the signature $s = \sum_{j=1}^{t} s_j\lambda_j(0) \pmod q$,
- computes its share $x_{n+1} = \sum_{j=1}^{t} \tilde{x}_{n+1}^{(j)} \pmod q$.

Fig. 3. TS-Sch admission protocol.

5. P_{n+1} first computes the signature s and its share x_{n+1} such that $s = \sum_{j=1}^{t} s_j \cdot \lambda_j(0) \pmod q$ and $x_{n+1} = \sum_{j=1}^{t} \tilde{x}_{n+1}^{(j)} \pmod q$. P_{n+1} verifies the signature (e,s) and x_{n+1} using normal Schnorr verification and VSS, respectively. If it succeeds, P_{n+1} creates its membership certificate GMC_{n+1} which contains $m = \{id_{n+1}, PK_{n+1}, \text{etc.}\}$ and (e,s). if verification fails, P_{n+1} traces s_j and $\tilde{x}_{n+1}^{(j)}$ using techniques presented in the following:
   (a) *Partial Share Tracing.* Correctness of partial secret share $\tilde{x}_{n+1}^{(j)}$ can be checked using VPSS.
   (b) *Partial Signature Tracing.* Correctness of individual partial signature s_j is verified by: $g^{s_j} = r_j y_j^e \pmod p$, where $y_j = g^{x_j}$ is calculated using the witnesses for VSS as in [5].

6.4. Membership authentication

Same as described in Section 5.4, except that the signature on the certificate is verified using Schnorr signature verification.

7. Threshold BLS based access control

We now describe the access control mechanism (referred to as TS-BLS[10]) based on the threshold BLS [4] scheme of [2].

7.1. BLS signature scheme

Boneh et al. [4] proposed a short signature scheme (referred to as BLS). BLS uses the system-wide parameters $(p, \mathbb{F}_p, a, b, P, q)$ based on Elliptic Curves (EC). The curve is represented by a equation: $y^2 = x^3 + ax + b$. $\mathbb{G}_1$ is set to be a group of order q generated by P, $\mathbb{G}_2$ is a subgroup of $\mathbb{F}_{p^2}^*$ of order q, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is defined to be a public *bilinear mapping*, satisfying $\hat{e}(aP, bQ) = \hat{e}(P,Q)^{ab}$ and non-degeneracy, $\hat{e}(P,P) \neq> 1$ for all $a, b \in \mathbb{Z}_q^*$ and $P, Q \in \mathbb{G}_1$. Also, $H_1 : \{0,1\}^* \to \mathbb{G}_1$ is the hash function that maps binary strings to non-zero points in $\mathbb{G}_1$. All of this information is published. In brief, the BLS signature scheme operates as follows:

1. *Key generation.* Pick random $x \in \mathbb{Z}_q^*$ and compute $Q = xP$. x is the private key and Q is the corresponding public key.
2. *Signing.* To sign a message m, compute $s = xH_1(m)$, where H_1 is a special hash function that maps binary strings onto points in $\mathbb{G}_1$. s is the signature on m.
3. *Verification.* Given (P, Q, m, s), check if $\hat{e}(Q, H_1(m)) = \hat{e}(P,s)$.

7.2. Bootstrapping

Similar to TS-DSA and TS-Sch, TS-BLS can be initialized by either: (1) a trusted dealer or (2) a group of $(2t-1)$ or more founding members. The bootstrapping procedure is exactly the same as in TS-DSA. Only difference is that the VSS operations are performed in the elliptic curve domain and threshold BLS signing is used to issue certificates.

7.3. Member admission

Let n ($\geqslant 2t-1$) be the number of current group members. The protocol steps are described below and Fig. 4 shows the protocol message flow.

1. Same as the step (1) in Section 5.3.
2. Group members who participate in admission reply with their respective GMC_is to P_{n+1} along with its signature.
3. P_{n+1} picks t out of t' ($\geqslant t$) sponsors, forms a list SL_{n+1} which contains the *id*s of t sponsors, signs it, and sends it to each P_j.
4. Each sponsoring member computes the partial signature s_j and the shuffled partial share of the secret $\tilde{x}_{n+1}^{(j)}$ such that $s_j = x_j H_1(m)$ and $\tilde{x}_{n+1}^{(j)} = x_j \lambda_j(id_{n+1}) + R_j \lambda_j(0) \pmod q$. Note that, unlike TS-DSA and TS-Sch, s_j is computed without Lagrange coefficient $\lambda_j(0)$ which means that TS-BLS signing does not require any interaction among t sponsoring members.
5. P_{n+1} first computes the signature s and its share x_{n+1} such that $s = \sum_{j=1}^{t} s_j \lambda_j(0) = \sum_{j=1}^{t} (x_j \lambda_j(0)) H_1(m) = xH_1(m)$ and $x_{n+1} = \sum_{j=1}^{t} \tilde{x}_{n+1}^{(j)} \pmod q$. P_{n+1} verifies the signature s using normal BLS verification. Also, P_{n+1} verifies x_{n+1} using the following ECC-version of VSS protocol such that $x_{n+1}P = \sum_{i=0}^{t-1} (id_{n+1})^i W_i$. If it succeeds, P_{n+1} creates its membership certificate GMC_{n+1} which contains $m = \{id_{n+1}, PK_{n+1}, \text{etc.}\}$ and s. If verification fails, P_{n+1} traces s_j and $\tilde{x}_{n+1}^{(j)}$ using techniques presented in the following:
   (a) *Partial share tracing.* Correctness of partial secret share $\tilde{x}_{n+1}^{(j)}$ can be checked using

[10] The identity-based version of this scheme appeared in [38].

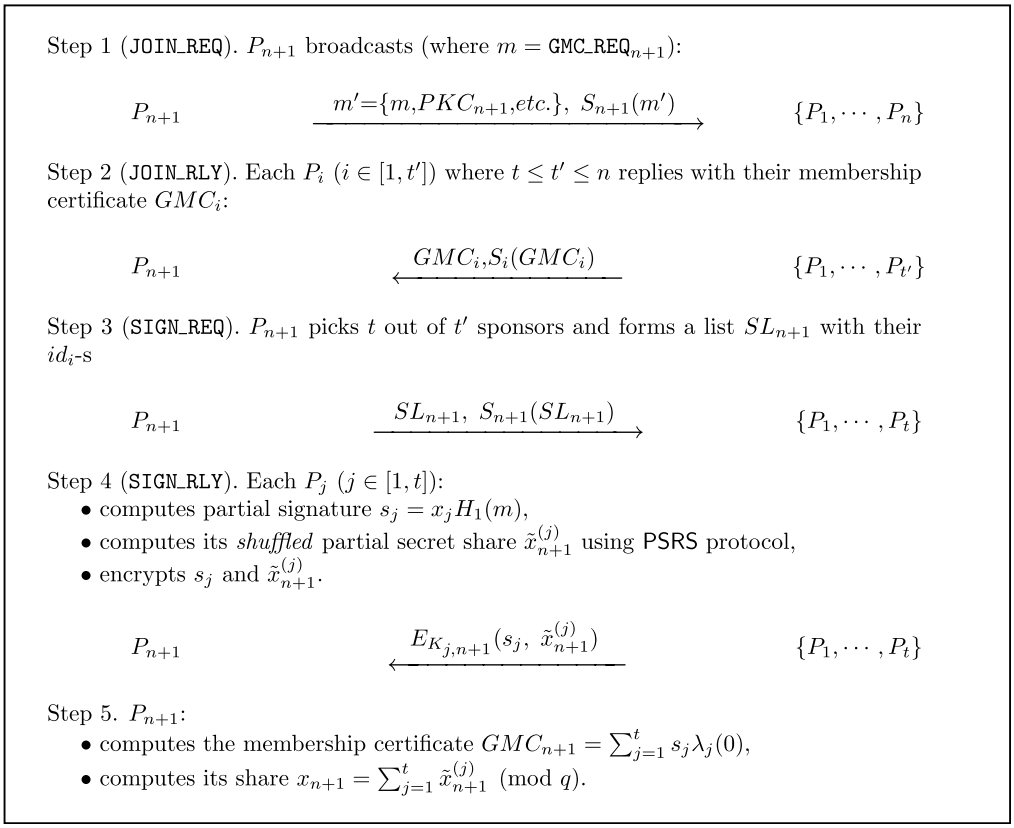


Fig. 4. TS-BLS admission protocol.

the following ECC-version of the VPSS technique: $\tilde{x}_{n+1}^{(j)}P = \lambda_j(id_{n+1})\sum_{i=0}^{t-1}(id_j)^i W_i + \lambda_j(0)R_jP$.

(b) *Partial signature tracing.* Correctness of each partial signature s_j can be verified by $\hat{e}(s_j,P) = \hat{e}(H_1(m),\lambda_j(0)\sum_{i=0}^{t-1} id_j^i W_i)$.

7.4. Membership authentication

Same as described in Section 5.4, except that the signature on the certificate is verified using BLS signature verification.

8. Performance evaluation

In this section, we compares the three access control mechanisms, TS-DSA, TS-Sch, and TS-BLS, in terms of their respective key features and their performance.

8.1. Complexity comparison

Table 2 summarizes the key features of each mechanism. In TS-DSA, $(2t-1)$ signers are required to tolerate $(t-1)$ faults, while t partial shares are needed to reconstruct the secret share for joining. Both TS-Sch and TS-BLS schemes require t partial signatures as well as t partial shares. Thus, to complete admission protocol, the group population should be at least $(2t-1)$ in TS-Sch and TS-BLS protocols, and $(3t-2)$ in TS-DSA. Both TS-DSA and TS-Sch require, by construction, the generation of a random value (and in turn interaction) among sponsors, while TS-BLS has a *fully non-interactive* signature generation.

Table 3 compares computation and communication costs of the three protocols[11]. As with computation costs, for admission, TS-DSA and TS-Sch require $O(t^2)$ exponentiations. TS-BLS requires $O(t^2)$ $\mathscr{M}$ operations (which are computationally equivalent to modula exponentiations in finite fields) and 2 $\mathscr{P}$ operations. For traceability, TS-DSA and TS-Sch schemes require $O(t^2)$ exponentiations, while TS-BLS requires $O(t^2)$ $\mathscr{M}$ and $2t$ $\mathscr{P}$ operations. This means that TS-DSA and TS-Sch should perform better than TS-BLS as far as

[11] The costs required for protecting each protocol message are not taken into account since these costs vary with the specific signature scheme.

Table 2
Feature comparison

Key features	TS-DSA	TS-Sch	TS-BLS
Security (for admission)	DL	DL	EC-DL
Minimum group size	$3t-2$	$2t-1$	$2t-1$
DoS resistance	Yes	Yes	Yes
Interaction among sponsors required	Yes	Yes	No
PSRS required	Yes	Yes	Yes

traceability is concerned. For membership authentication, both TS-DSA and TS-Sch requires 2 exponentiations. On the other hand, TS-BLS requires 2 $\mathscr{P}$ operations.

In terms of overall communication costs, all protocols require at least $O(t^2)$ unicasts and consume $O(t^2 \log q + t \log p)$ bits. However, we observe that TS-DSA requires significantly more rounds and higher bandwidth.

8.2. Experimental setups

We now describe the experimental testbeds for measuring the performance of our proposed protocols. We ran experiments in a *real* wireless MANET environment and also measured energy costs for each scheme with power measuring system below.

8.2.1. Wireless mobile ad hoc networks

We used five laptop computers for our wireless experimental set-up: four laptop computers with Pentium-3 800 MHz CPU and 256 MB memory and one laptop computer with Mobile Pentium 1.8 GHz CPU and 512 MB memory. Each machine is configured with 802.11*b* in ad hoc mode and runs the Optimized Link State Routing Protocol (OLSR) [29]. Each machine runs Linux kernel 2.4. To simulate more than five nodes using just five laptops, we ran more than one client process on each laptop.

8.2.1.1. Effect of mobility. Node mobility certainly affects the performance of our access control mechanisms. Clearly, if nodes move around (and become unreachable to each other and to the joining node) while the protocol is in execution, the protocol might not terminate successfully. On the other hand, mobility can help the joining node to move to a new, more dense location, where it has enough neighbors to sponsor its admission.

In our experimental evaluation to follow, we did not include the affect of mobility due to the following reason: our evaluation measures the protocol (computation, communication, power) overhead under the assumption that the protocol successfully terminates, i.e., under the assumption that the joining node has enough online neighbors available for the duration the protocol is executed. In other words, we assume that the nodes who sponsor admission do not move around (and become unreachable to each other and to the joining node) during the execution of the protocol.

8.2.2. Power measurement systems

To measure consumption of battery power, we configured the following equipment, as shown in Fig. 5. The test machine was an iPAQ (model H5555) running Linux (Familiar-0.7.2). The CPU

Table 3
Computation and communication complexities

Category			TS-DSA	TS-Sch	TS-BLS
Computation	Admission	$\mathscr{E}$	$5t^2+6t$	t^2+2t+3	N/A
		$\mathscr{M}$	N/A	N/A	$2t^2+t+1$
		$\mathscr{P}$	N/A	N/A	2
	Traceability	$\mathscr{E}$	$3t^2+9t-3$	$2t^2+5t$	N/A
		$\mathscr{M}$	N/A	N/A	$2t^2+3t$
		$\mathscr{P}$	N/A	N/A	$2t$
	Membership Authentication	$\mathscr{E}$	2	2	N/A
		$\mathscr{M}$	N/A	N/A	3
		$\mathscr{P}$	N/A	N/A	2
Communication	Round	Broadcast	1	1	1
		Unicast	$4t^2$	t^2+2t	t^2+2t
	Bandwidth	$\log q$-bit	$5t^2+4t-3$	$2t^2+3t$	$2t^2+2t$
		$\log p$-bit	$14t-7$	$6t$	$3t$

$\mathscr{E}$: modular exponentiation, $\mathscr{M}$: scalar-point-multiplication in ECC, $\mathscr{P}$: Tate pairing operation in ECC.

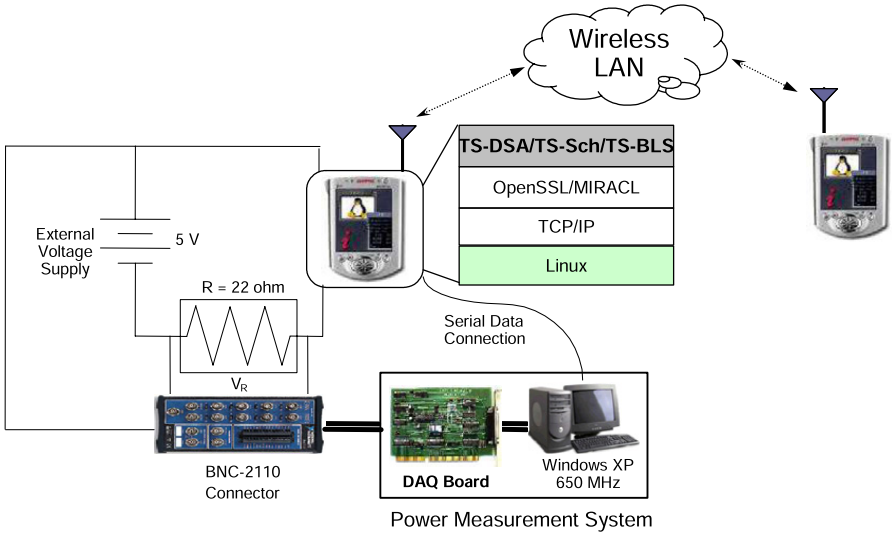


Fig. 5. Power measurement testbed.

on the iPAQ is a 400 MHz Intel XScale with 48MB of flash memory and 128MB of SDRAM. In order to obtain accurate power measurements, we removed the battery from the iPAQ during the experiment and placed a resistor in series with power supply. We used a National Instruments PCI DAQ (Data AcQuisition) board to sample the voltage drops across the resistor to calculate the current at 1000 samples per second.

8.3. Test methodology

1. *Parameter selection.* To perform fair comparisons, the size of the parameter q was set to be 160-bit and p to be 1024-bit except for TS-BLS. For TS-BLS experiments, we used the elliptic curve E defined by the equation: $y^2 = x^3 + 1$ over $\mathbb{F}_p$ with $p > 3$ a prime satisfying $p = 2 \pmod 3$ and q being a prime factor[12] of $p + 1$. The parameter p is a 512-bit prime in order to make sure that the security of pairing $\hat{e}$ is equivalent to the security as in finite field of 1024 bits.[13] The measurements were performed with different threshold values t from 1 to 9. We used 1024-bit RSA signature with the fixed public exponent 65537 ($=2^{16}+1$) for protocol message authentication. All experiments were repeated 1000 times for each measurement in order to get fairly accurate average results.[14]
2. *Test cases.* We measured the respective costs of admission, energy consumption (of admission), traceability, and membership authentication.

8.4. Basic operations

We first present the costs of the primitive operations in Table 4. For measuring the costs of basic operations, we used a machine with Mobile Pentium 1.8 GHz CPU and 512 MB memory.

8.5. Experimental results

We now present and discuss the performance measurement results for the proposed TS-DSA, TS-Sch and TS-BLS schemes.

8.5.1. Admission results

Fig. 6a shows the admission cost with varying threshold for TS-DSA, TS-Sch and TS-BLS schemes. The admission costs also include the verification of the membership certificate and the secret share.

As shown in Fig. 6a, TS-BLS exhibits appreciably better performance that TS-DSA. The results imply that the amount of communication in TS-DSA

[12] By Euler's theorem, q must divide $\#E(\mathbb{F}_p)$. For the curve $y^2 = x^3 + 1$, $\#E(\mathbb{F}_p) = p + 1$.

[13] The $\mathbb{G}_1$ is a subgroup of points generated by P such that $P \in E(\mathbb{F}_p)$. The $\mathbb{G}_2$ is a subgroup of $\mathbb{F}^*_{p^2}$ of order q. The bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is the well-known Tate pairing. Computing discrete log in $\mathbb{F}_{p^2}$ is sufficient for computing discrete log in $\mathbb{G}_1$. Therefore, for proper security of discrete log in $\mathbb{F}_{p^2}$ the prime p should be at least 512-bits long (so that the group size is at least 1024-bits long).

[14] The source code is publicly available at [32].

Table 4
Costs of primitive operations (Mobile Pentium 1.8 GHz)

Function		Modulus (bits)	Exponent (bits)	Average time (ms)
Signing	DSA	1024	160	2.58
	Schnorr	1024	160	2.20
	BLS	512	160	4.81
Verification	DSA	1024	160	4.17
	Schnorr	1024	160	4.20
	BLS	512	160	27.05

contributes significantly to the overall cost of admission, although computation-wise it is still quite efficient (see Table 3).

8.5.2. Energy consumption results

This experiment is quite tricky to measure fairly. The energy consumption is directly proportional to the processing time and thus it is meaningless to measure energy consumption with all the test cases above. However, it is well known that, in many small devices such as low-end MANET nodes or sensors, sending a single bit is roughly equivalent to performing 1000 32-bit computations in terms of batter power consumption [1]. Therefore, we measured power consumption in terms of communication bandwidth required by each admission protocol. For more details, we sent some bulk data (e.g., 100 Mbytes) from a single iPAQ PDA (refer to Fig. 5), measured power consumed while sending out this data, and then computed the average power consumption per bit. After that, we calculated power consumption of each admission protocol by multiplying this measurement result by the bit length of the transmitted data.

Energy consumption results are plotted in Fig. 6b. These results clearly illustrate that TS-BLS is the most energy-efficient, since they require the smallest amount of bandwidth amongst the respective admission protocols.

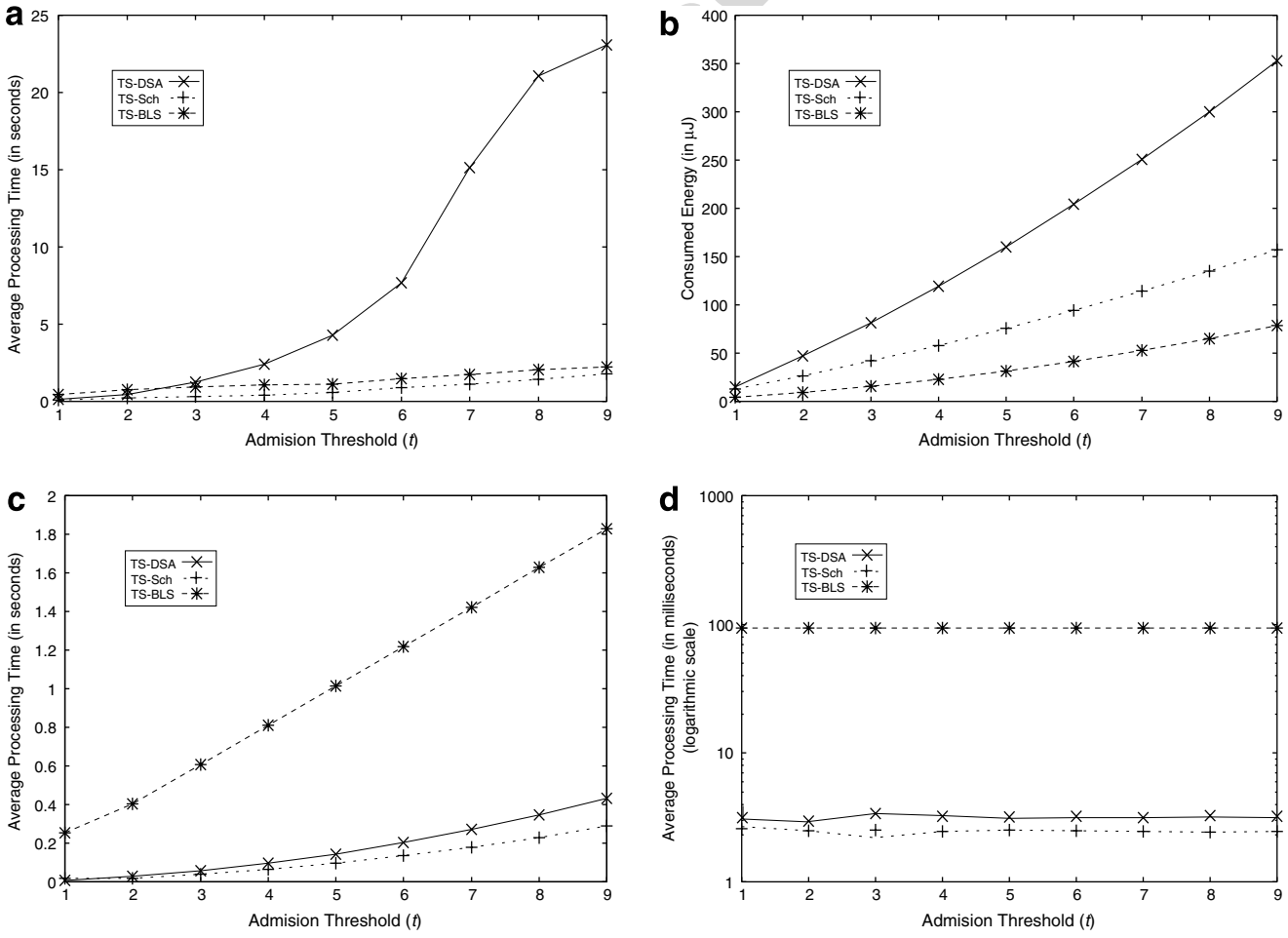


Fig. 6. Cost comparison: (a) member admission, (b) energy consumption, (c) traceability and (d) membership authentication.

8.5.3. Traceability results

Traceability costs are presented in Fig. 6c. Due to the costly computation of Tate pairings, TS-BLS performs poorly, as compared to TS-DSA and TS-Sch. However, since the misbehavior in the admission protocol leads ultimately to the eviction of the corresponding group member, we argue that traceability is a *rare exceptional* measure; thus we consider its costs to be relatively unimportant.

8.5.4. Membership authentication results

The costs of membership authentication are shown in Fig. 6d. The results show that costs for TS-DSA and TS-Sch are very close to each other and relatively constant while TS-BLS cost is very high due to expensive pairing operation.

9. Conclusion

In this paper, we explored the utility of various existing threshold signature schemes in building distributed access control mechanisms for ad hoc groups. We first showed that none of the threshold (or proactive) RSA signature schemes in the literature are applicable for our purpose. Next, we implemented three access control mechanisms based on discrete-logarithm based threshold signatures, threshold DSA (TS-DSA), threshold Schnorr (TS-Sch) and threshold BLS (TS-BLS), and evaluated them in a real MANET setting. Based on our evaluation, we conclude that overall TS-Sch is the most efficient mechanism, followed by TS-BLS and TS-DSA.

Appendix A. Cryptographic primitives

A.1. Threshold secret sharing (TSS)

In this section, we present Shamir's secret sharing scheme [41] which is based on polynomial interpolation. We will refer to it as TSS. To distribute shares of a secret x among n entities, a trusted dealer TD chooses a polynomial $f(z)$ over $\mathbb{Z}_q$ of degree $(t-1)$: $f(z) = \sum_{i=0}^{t-1} a_i z^i \pmod q$, where the constant term a_0 is set to the group secret x; $f(0) = a_0 = x$. TD computes each entity's share x_i such that $x_i = f(id_i)$, where id_i is an identifier of entity P_i, and securely transfers x_i to P_i. Note that after distributing at least t secret shares, the dealer is no longer required.

Then, any group of t entities who have their shares can recover the secret using the Lagrange interpolation formula: $f(z) = \sum_{i=1}^{t} x_i \lambda_i(z) \pmod q$, where $\lambda_i(z) = \prod_{j=1, j\neq i}^{t} \frac{z-id_j}{id_i-id_j} \pmod q$. Since $f(0) = x$, the shared secret may be expressed as: $x = f(0) = \sum_{i=1}^{t} x_i \ \lambda_i(0) \pmod q$. Thus, the secret x can be recovered only if at least t shares are combined. In other words, no coalition of less than t entities yields any information about x.

A.2. Joint secret sharing (JSS)

This scheme (due to Pederson [31]), denoted by JSS, extends Shamir's secret sharing by removing the need for a centralized dealer to choose a polynomial and distribute shares. In this scheme, the entities 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 each entity P_i over $\mathbb{Z}_q$.

Suppose there are n entities in a system $(P_1, \ldots, P_n)$. It will be assumed that all entities of the group have previously agreed on the prime q. Each P_i chooses at random a polynomial $f_i(z) \in \mathbb{Z}_q$ of degree $(t-1)$ such that $f_i(0) = r_i$ where r_i is a random secret that P_i selects. Let $f_i(z) = a_{i0} + a_{i1}z + \cdots + a_{i,t-1}z^{t-1} \pmod q$, where $a_{i0} = r_i$. P_i computes P_j's share $\hat{x}_j^{(i)} = f_i(id_j)$ for P_j $(j \in [1,n])$, and *securely* sends it to P_j (in particular P_i keeps $\hat{x}_i^{(i)}$). Note that the share values should be transmitted over the secure channel. P_j computes its share x_j of the secret x as the sum of all shares received: $x_j = \sum_{i=1}^{n} \hat{x}_j^{(i)}$.

Let $f(z)$ denote the combined polynomial over $\mathbb{Z}_q$. It is given by: $f(z) = f_1(z) + \cdots + f_n(z) \pmod q$. By construction $x_j = f(id_j)$ for $j \in [1,n]$, and therefore x_j is a share of x such that $x_j = f(id_j) = \sum_{i=1}^{n} f_i(id_j) = \sum_{i=1}^{n} \hat{x}_j^{(i)}$ and $x = f(0) = \sum_{i=1}^{n} f_i(0) = \sum_{i=1}^{n} a_{i0} = \sum_{i=1}^{n} r_i \pmod q$. Once every entity has its own share, any coalition of t entities can jointly recover the secret x using Lagrange interpolation as in TSS in Appendix A.1.

A.3. Joint zero secret sharing (JZSS)

This scheme, which first appeared in [15], is a variant of joint secret sharing where the shared secret is zero. In other words, this scheme is the same as JSS except that in first step, each entity picks a random $(t-1)$-degree polynomial $f_i(z) \in \mathbb{Z}_q$ such that $f_i(0) = 0$. We refer to it as a *Joint Zero Secret Sharing*, denoted by JZSS. It is used in proactive

secret sharing and partial share random shuffling for re-randomizing a secret share.

A.4. Verifiable secret sharing (VSS)

If we suppose that some entities 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* [8], denoted by VSS, 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 *TD* to distribute the shares is the same as in Appendix A.1. VSS is achieved by the following procedure:

1. *Witness generation.* The *TD* randomly selects a polynomial $f(z) = \sum_{i=0}^{t-1} a_i z^i$, computes secret shares x_i, and transfers them to each entity securely. Also, *TD* chooses an element $g \in \mathbb{Z}_p^*$ of order q, and computes W_i, for $i \in [0, t-1]$, called *witness*, such that $W_i = g^{a_i}$. Then, *TD* publishes these W_is in some public domain (e.g., a directory server).[15]
2. *Share verification.* When each entity P_i receives its share x_i, it verifies x_i by checking: $g^{x_i} = \prod_{j=0}^{t-1} [W_j]^{(id_i)^j} \pmod p$.

A.5. Partial secret sharing (PSS)

As a result of secret sharing, each honest entity P_i obtains a secret share x_i. Then, the share x_{n+1} for a prospective entity P_{n+1} can be computed through collaboration of t existing entities in the group when the *TD* is no longer available. We call this a *Partial Secret Share*, referring it to as PSS. P_{n+1} receives t partial shares $x_{n+1}^{(j)}$s from a set of t entities called *sponsors*. It will be assumed that the t number of indices, j $(= 1, \ldots, t)$, are given to each P_j by P_{n+1}. The details are as follows: each P_j computes a partial secret share for P_{n+1} as: $x_{n+1}^{(j)} = x_j \cdot \lambda_j(id_{n+1}) \pmod p$, where x_j is P_j's own secret share. Then, P_j securely sends $x_{n+1}^{(j)}$ to P_{n+1}. Given t partial shares and an identity of P_{n+1}, the secret share x_{n+1} can be computed: $x_{n+1} = \sum_{j=1}^{t} x_{n+1}^{(j)} \left[= \sum_{j=1}^{t} x_j \cdot \lambda_j(id_{n+1}) \right] \pmod q$. Recall that $f(z) = \sum_{i=1}^{t} x_i \lambda_i(z)$ and $x_{n+1} = f(id_{n+1})$.

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

A.6. Partial share random shuffling (PSRS)

In PSS the above, P_{n+1} needs to be provided (in a distributed manner) with its share x_{n+1} of the group secret x. However, in case that each sponsor P_j issues P_{n+1} a partial secret share $x_{n+1}^{(j)}$ such that $x_{n+1}^{(j)} = x_j \lambda(id_{n+1})$, P_{n+1} (or an adversary who corrupts P_{n+1}) can easily recover each x_j and in turn the group secret x, since Lagrange coefficients $\lambda_j(id_{n+1})$ are *publicly known*, P_{n+1} can obtain x_j by dividing $x_{n+1}^{(j)}$ by $\lambda_j(id_{n+1})$. To remedy this, P_js must randomize the issued partial shares. We call this procedure as *Partial Share Random Shuffling*, denoted by PSRS.

The detailed procedure is as follows. All of the t sponsors perform the JZSS, as in Appendix A.3, by setting the constant term of their respective polynomials to zero. Also, the witness values of the polynomials are broadcast to enable VSS. At the end of JZSS, every P_j possesses a random share R_j of the shared secret zero. Now, P_j provides the *shuffled* partial secret share $\tilde{x}_{n+1}^{(j)}$ for P_{n+1} : $\tilde{x}_{n+1}^{(j)} = x_{n+1}^{(j)} + R_j \lambda_j(0) = x_j \lambda_j(id_{n+1}) + R_j \lambda_j(0) \pmod q$. Since $\sum_{j=1}^{t} R_j \lambda_j(0) = 0$, P_{n+1}'s secret share x_{n+1} is given by $x_{n+1} = \sum_{j=0}^{t-1} \tilde{x}_{n+1}^{(j)}$.

A.7. Verifiable partial secret sharing (VPSS)

The idea of VSS can be easily extended to verify correctness of partial shares that P_{n+1} receives from sponsors. We call this a *Verifiable Partial Secret Sharing*, referred to as VPSS. Since $\tilde{x}_{n+1}^{(j)} = x_j \lambda_j(id_{n+1}) + R_j \lambda_j(0)$ where R_j is the random number for the PSRS technique as explained above, to check if $\tilde{x}_{n+1}^{(j)}$ is correctly computed and trace which of the P_js sent back false values, if any, following equation is verified: $g^{\tilde{x}_{n+1}^{(j)}} = \left[\prod_{k=0}^{t-1} (W_k)^{id_j^k} \right]^{\lambda_j(id_{n+1})} g^{R_j \lambda_j(0)} \pmod p$, where g^{R_j} is computed using the broadcast witness values of the shared polynomial among the sponsors.

Appendix B. Threshold RSA scheme of [23,21,20,24,22]

A *TD* is involved in a one-time setup to bootstrap the system. *TD* generates the standard RSA private/public key pair, i.e., it picks two random primes p and q, sets $N = pq$, sets (e, N) as a public key where $gcd(e, N) = 1$, and as a private key it sets a number $d < N$ s.t. $ed = 1 \bmod \phi(N)$. Once the standard

$$
\begin{array}{lll}
P_{n+1} \rightarrow P_i: & m, S_{n+1}(m), PKC_{n+1} & (1)\\
P_{n+1} \leftarrow P_i: & GMC_i & (2)\\
P_{n+1} \rightarrow P_j: & SL_{n+1} & (3)\\
P_{n+1} \leftarrow P_j: & s_j, \tilde{x}_{n+1}^{(j)} & (4)
\end{array}
$$

$$s_j = (m)^{d_j} \pmod N$$

Fig. 7. TS-RSA admission protocol.

RSA key pair is chosen, *TD* secret-shares the RSA secret key d using a slight modification of TSS. Namely, *TD* selects a random polynomial $f(z) = a_0 + a_1 z + \cdots + a_t z^t$ over $\mathbb{Z}_N$ of degree t, such that the group secret is $f(0) = d \pmod N$. Next, *TD* gives to each member P_i, for $i = 1, \ldots, n$, a secret share $x_i = f(id_i) \pmod N$. Notice that the secret d is shared over a public composite modulus N as opposed to a prime modulus as in the original scheme of Shamir and a secret modulus $\phi(N)$ in Shoup's scheme. Since there may be compromised members who can generate false shares and false signatures thereafter, the dealer provides a *witness* of $f(z)$ which is represented by $\{g^{a_0}, g^{a_1}, \ldots, g^{a_{t-1}}\} \pmod N$ for a certain $g \in \mathbb{Z}_N^*$, and publishes it for VSS [8].

Fig. 7 shows the message flows for the admission protocol. Each member P_j, for $id_j \in SL_{n+1}$, outputs its partial signature s_j on m as $s_j = m^{d_j} \pmod N$, where $d_j = x_j \lambda_j(0) \pmod N$. In addition, P_j also provides P_{n+1} with its partial secret share $\tilde{x}_{n+1}^{(j)}$ (computed over modulus N) after shuffling using PSRS. P_{n+1} reconstructs the RSA signature using the *t-bounded offsetting* algorithm (refer to [21] for details) and its secret share x_{n+1} such that $x_{n+1} = \sum_{id_j \in SL_{n+1}} \tilde{x}_{n+1}^{(j)} \pmod N$.

References

[1] K. Barr, K. Asanovic, Energy aware lossless data compression, in: International Conference on Mobile Systems, Applications, and Services (MobiSys), May 2003.

[2] A. Boldyreva, Efficient threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme, in: Proceedings of International Workshop on Practice and Theory in Public Key Cryptography, LNCS, vol. 2567, 2003, pp. 31–46.

[3] D. Boneh, M. Franklin, Efficient generation of shared RSA key, in: CRYPTO'97, LNCS, vol. 1294, 1997, pp. 425–439.

[4] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, in: C. Boyd (Ed.), ASIACRYPT'01, LNCS, vol. 2248, 2001, IACR, pp. 514–532.

[5] C. Castelluccia, N. Saxena, J.H. Yi, Self-configurable key pre-distribution in mobile ad hoc networks, in: IFIP Networking Conference, May 2005.

[6] Y. Desmedt, Y. Frankel, Threshold Cryptosystems, in: CRYPTO'89, LNCS, vol. 435, 1990, pp. 307–315.

[7] T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: IEEE Transactions on Information Theory, vol. 31 (July), 1985, pp. 469–472.

[8] P. Feldman, A practical scheme for non-interactive verifiable secret sharing, in: 28th Symposium on Foundations of Computer Science (FOCS), 1987, pp. 427–437.

[9] Y. Frankel, P. Gemmell, P.D. MacKenzie, M. Yung, Proactive RSA, in: CRYPTO'97, LNCS, vol. 1294 (August), 1997, pp. 440–454.

[10] Y. Frankel, P.D. MacKenzie, M. Yung. Adaptive security for the additive-sharing based proactive RSA, in: Public Key Cryptography 2001, LNCS, vol. 1992, 2001, pp. 240–263.

[11] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Robust threshold DSS signatures, in: EURO-CRYPT'96, LNCS, vol. 1070, 1996, pp. 354–371.

[12] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Secure applications of Pedersen's distributed key generation protocol, in: RSA Conference – The Cryptographers' Track, April 2003.

[13] M.T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, E. Uzun, Loud and Clear: Human-verifiable authentication based on audio, in: International Conference on Distributed Computing Systems (ICDCS), July 2006. Available from: <http://www.ics.uci.edu/ccsp/lac>.

[14] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, Proactive public key and signature systems, in: ACM Conference on Computers and Communication Security, 1997.

[15] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, Proactive secret sharing, or how to cope with perpetual leakage, in: CRYPTO'95, LNCS, vol. 963, 1995, pp. 339–352.

[16] R. Housley, W. Polk, W. Ford, D. Solo, Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, RFC 3280, IETF, April 2002.

[17] Y.-C. Hu, A. Perrig, D. B. Johnson, Ariadne: a secure on-demand routing protocol for ad hoc networks, in: Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (Mobicom 2002), September 2002.

[18] S. Jarecki, N. Saxena, J.H. Yi, An attack on the proactive RSA signature scheme in the URSA ad hoc network access control protocol, in: ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), October 2004, pp. 1–9.

[19] Y. Kim, D. Mazzocchi, G. Tsudik, Admission control in peer groups, in: IEEE International Symposium on Network Computing and Applications (NCA), April 2003.

[20] J. Kong, H. Luo, K. Xu, D.L. Gu, M. Gerla, S. Lu, Adaptive security for multi-level ad-hoc networks, Journal of Wireless Communications and Mobile Computing (WCMC) 2 (2002) 533–547.

[21] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, Providing robust and ubiquitous security support for MANET, in: IEEE 9th International Conference on Network Protocols (ICNP), 2001, pp. 251–260.

[22] H. Luo, J. Kong, P. Zerfos, S. Lu, L. Zhang, URSA: ubiquitous and robust access control for mobile ad hoc networks, in: IEEE/ACM Transactions on Networking (ToN), December 2004.

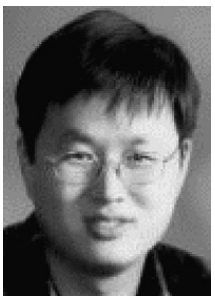
[23] H. Luo, S. Lu, Ubiquitous and robust authentication services for ad hoc wireless networks, Technical Report TR-200030, Department of Computer Science, UCLA, 2000. Available online at <http://citeseer.ist.psu.edu/luo00ubiquitous.html>.

[24] H. Luo, P. Zerfos, J. Kong, S. Lu, L. Zhang, Self-securing ad hoc wireless networks, in: Seventh IEEE Symposium on Computers and Communications (ISCC'02), 2002.

[25] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of applied cryptography, CRC Press series on discrete mathematics and its applications, 1997, ISBN 0-8493-8523-7.
[26] M. Narasimha, G. Tsudik, J.H. Yi, On the utility of distributed cryptography in P2P and MANETs: the case of membership control, in: IEEE International Conference on Network Protocol (ICNP), November 2003, pp. 336–345.
[27] NIST, Digital Signature Standard. Technical Report 169, August 1991.
[28] K. Ohta, S. Micali, L. Reyzin, Accountable subgroup multisignatures, in: ACM Conference on Computer and Communications Security, November 2001, pp. 245–254.
[29] OLSR Protocol, <http://menetou.inria.fr/olsr>.
[30] R. Ostrovsky, M. Yung, How to withstand mobile virus attacks, in: 10th ACM Symp. on the Princ. of Distr. Comp., 1991, pp. 51–61.
[31] T.P. Pedersen, A threshold cryptosystem without a trusted party, in: D. Davies (Ed.), EURO-CRYPT'91, LNCS, vol. 547, 1991, IACR, pp. 552–526.
[32] Peer Group Admission Control Project. <http://sconce.ics.uci.edu/gac>.
[33] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, Spins: security protocols for sensor networks, in: Mobile Computing and Networking, 2001.
[34] D. Pointcheval, J. Stern, Security proofs for signature schemes, in: EUROCRYPT'96, LNCS, vol. 1070, May 1996, pp. 387–398.
[35] T. Rabin, A simplified approach to threshold and proactive RSA, in: CRYPTO'98, LNCS, vol. 1462, 1998, pp. 89–104.
[36] N. Saxena, J.-E. Ekberg, K. Kostiainen, N. Asokan, Secure device pairing based on a visual channel (short paper), in: IEEE Symposium on Security and Privacy (ISP'06), May 2006.
[37] N. Saxena, G. Tsudik, J.H. Yi, Admission control in peer-to-peer: design and performance evaluation, in: ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), October 2003, pp. 104–114.
[38] N. Saxena, G. Tsudik, J.H. Yi, Identity-based access control for ad-hoc groups, in: International Conference on Information Security and Cryptology (ICISC), December 2004.
[39] C.P. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology 4 (3) (1991) 161–174.
[40] B. Schoenmakers, A simple publicly verifiable secret sharing scheme and its application to electronic voting, in: M. Wiener (Ed.), CRYPTO'99, LNCS, 1666, 1999, IACR, pp. 148–164.
[41] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.
[42] V. Shoup, Practical Threshold Signatures. In EURO-CRYPT'00, LNCS, vol. 1807, 2000, pp. 207–220.
[43] M. Steiner, G. Tsudik, M. Waidner, Cliques: a new approach to group key agreement, IEEE Transactions on Parallel and Distributed Systems (August) (2000).
[44] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups, IEEE Transactions on Parallel and Distributed Systems (July) (2000).
[45] L. Zhou, Z.J. Haas, Securing ad hoc networks, IEEE Network Magazine 13 (6) (1999) 24–30.
[46] L. Zhou, F. Schneider, R. van Renesse, COCA: a secure distributed on-line certification authority, ACM Transactions on Computer Systems 20 (4) (2002) 329–368.

Nitesh Saxena is an Assistant Professor in the department of Computer and Information Science at Polytechnic University, starting Fall 2006. He obtained his Ph.D. in Information and Computer Science from University of California, Irvine, in summer 2006. He holds an M.S. degree in Computer Science from UC Santa Barbara, and a Bachelor's degree in Mathematics and Computing from Indian Institute of Technology, Kharagpur, India. His research spans all areas of information security with core emphasis on network and distributed system security and applied cryptography. His Ph.D. dissertation entitled Decentralized Security Services'' has been nominated for the ACM Dissertation Award for the year 2006.

Gene Tsudik is a Professor of Computer Science at the University of California, Irvine. He has been conducting research active in internetworking, network security and applied cryptography since 1987. He obtained a Ph.D. in Computer Science from USC in 1991; his dissertation focused on access control in internetworks. Before coming to UC Irvine in 2000, he was a Project Leader at IBM Research, Zurich Laboratory (1991–1996) and USC Information Science Institute (1996–2000). Over the years, his research interests included: routing, firewalls, authentication, mobile/wireless network security, secure e-commerce, anonymity, secure group communication, digital signatures, key management, ad hoc network routing, and, more recently, database privacy and secure storage. Some of Professor Tsudiki s notable research contributions include: Inter-Domain Policy Routing (IDPR), IBM Network Security Program (KryptoKnight), IBM Internet Keyed Payment (iKP) protocols, Peer Group Key Management (CLIQUES) and Mediated Cryptographic Services (SUCSES). He has over 100 refereed publications and 7 patents. Since 2002 he has been serving as Associate Dean of Research and Graduate Studies in the Donald Bren School of Information and Computer Sciences at UCI. He is a member of the IEEE.



Jeong Hyun Yi is a principal researcher at Samsung Advanced Institute of Technology (SAIT). He received his Ph.D. in Information and Computer Science with his advisor, Dr. Gene Tsudik, from University of California, Irvine in 2005. He received his M.S. and B.S. in Computer Science at Soongsil University, Korea in 1995 and 1993, respectively. He was a senior researcher at Electronics and Telecommunication Research Institute (ETRI), Korea from 1995 to 2001 and a guest researcher at National Institute of Standards and Technology (NIST), MD, USA From 2000 to 2001. His research interests are in network security, applied cryptography, ubiquitous computing, RFID and wireless sensor networks.