DoS-Resistant Self-Keying Mobile Ad-Hoc Networks

Claude Castelluccia, Jeong H. Yi

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

DoS-Resistant Self-Keying Mobile Ad-Hoc Networks

Claude Castelluccia — Jeong. H. Yi

N° 5373

November 2004

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--5373--FR+ENG

DoS-Resistant Self-Keying Mobile Ad-Hoc Networks

Claude Castelluccia*, Jeong. H. Yi †

Thème 1 — Réseaux et systèmes
Projets Planete

Rapport de recherche n° 5373 — November 2004 — 28 pages

Abstract: We present a new scheme that allows two nodes of a Mobile Ad-hoc network to compute a shared key without communicating. Such service is important to secure routing protocols [1, 2, 3]. The scheme is based on the novel combination of two well-known techniques: key pre-distribution and threshold secret sharing. Each node only needs to store a small number of keys, independent of the network size.

The proposed scheme is secure against collusion of up to a certain number of nodes. Furthermore, it is robust and DoS-resistant since a node that joins a network can efficiently verify each share it obtains from so-called authorization nodes and trace invalid shares. We evaluate and compare – via analysis and experiments – the performance of the different stages of our scheme (node join, key derivation, verification and traceability) with the performance of the Threshold-DSA based scheme proposed in [4, 5]. Results clearly indicate that the new scheme is much more practical.

Key-words: Mobile Ad-hoc networks, key exchange, security

* INRIA Rhône-Alpes, PLANETE group
† Computer Science Department, UC Irvine

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

DoS-Resistant Self-Keying Mobile Ad-Hoc Networks

Résumé : Ce rapport présente un nouveau protocole qui permet à deux noeuds d'un réseau mobile ad-hoc de s'échanger une clé secrète sans communiquer. Ce service est important pour sécuriser les protocoles de routage [1, 2, 3]. Notre solution repose sur la combinaison innovante de deux techniques cryptographiques connues: un système de pré-distribution de clés et de partage de secret à seuil. Dans ce papier, nous évaluons - par analyses et expérimentations- les performances de notre proposition.

Mots-clés : réseaux ad-hoc, échange de clés, sécurité

DoS-Resistant Self-Keying Mobile Ad-Hoc Networks

Claude Castelluccia, Jeong Hyun Yi

We present a new scheme that allows two nodes of a Mobile Ad-hoc network to compute a shared key without communicating. Such service is important to secure routing protocols [1, 2, 3]. The scheme is based on the novel combination of two well-known techniques: key pre-distribution and threshold secret sharing. Each node only needs to store a small number of keys, independent of the network size.

The proposed scheme is secure against collusion of up to a certain number of nodes. Furthermore, it is robust and DoS-resistant since a node that joins a network can efficiently verify each share it obtains from so-called authorization nodes and trace invalid shares. We evaluate and compare – via analysis and experiments – the performance of the different stages of our scheme (node join, key derivation, verification and traceability) with the performance of the Threshold-DSA based scheme proposed in [4, 5]. Results clearly indicate that the new scheme is much more practical.

1 Introduction

Mobile Ad-hoc Networks (MANET-s) are, by their very nature, vulnerable to many types of attacks. The security of MANET-s is often predicated on the availability of efficient key management techniques. However, the usual features of: (1) lack of a centralized authority and (2) dynamic nature of MANET-s, represent major obstacles to providing secure, effective and efficient key management. What further complicates the issue is that, in many applications (such as secure routing [1, 2, 3]) cryptographic keys need to be established *prior* to communication. As a result, standard key exchange solutions, e.g., Station-to-Station protocol [6], are not appropriate since: (1) they require the nodes to interact and (2) they rely on some form of a Public Key Infrastructure (PKI) which is not usually available in MANET-s. Related to the latter is the underlying use of public key cryptography which is too expensive for some mobile devices.

Contributions: This paper proposes an efficient and secure key management solution for MANET-s. It results from a novel blending of two well-known techniques: key predistribution [7, 8] and threshold secret sharing techniques [9]. In this scheme, a node joins a

*INRIA Rhône-Alpes, PLANETE group
†Computer Science Department, UC Irvine

MANET, by receiving a secret token from each of t different authorization nodes, where t is a security parameter. (An authorization node is simply a member of the MANET which has some extra privileges.) The scheme is auto-configurable in the sense that a node becomes a member only if it is approved by at least t authorization nodes. Similarly, a current member can get promoted to an authorization node, if it is approved by at least t other authorization nodes. Once a node becomes member, it can compute a secret key with any other members without interaction. If a member is promoted to an authorization node, it can participate in admission of new members and authorization nodes. The proposed scheme is secure against collusion of upto a certain number of compromised nodes. Furthermore, the scheme includes verifiability and traceability procedures that allow a joining node to efficiently verify the validity of each secret token it obtains from authorization nodes. As a result, when a node receives a secret token it is able to verify that: (1) the issuing node is a valid authorization node, (2) the token itself is valid (this is important to protect against compromised authorization nodes), and (3) in case a token is invalid, its originator can be traced.

The contribution of this paper is not limited to just the design of an efficient key distribution scheme. We also demonstrate our claims of efficiency via extensive analysis and experiments. The scheme has been implemented and tested in a real MANET setting and its performances has been compared with that of a recent key distribution scheme which uses Threshold-DSA (TS-DSA) signature [4, 5]. We show that the new scheme is more practical, since it avoids interaction and involves significantly less computation.

Organization: The rest of this paper is organized as follows: Section 2 overviews the related work. Section 3 provides some background on necessary cryptographic building blocks. Section 4 proposes our *Threshold Key Predistribution (TKP)* scheme. Section 5 describes the protocol in detail. We argue the security of the proposed scheme in Section 6. Finally, in Section 7, we describe the implementation and the performance of our scheme and compare it with one of threshold signature scheme based on DSA.

2 Related Work

Key distribution can be easily achieved if we assume the existence of a PKI. However, this assumption is not realistic in many MANET environments. In their seminal paper, Zhou and Haas proposed to distribute a Certification Authority (CA) service among several nodes of the network [10]. They also suggested the use of threshold cryptography and proactive secret sharing to improve the security and robustness of this service. They observed that the use of threshold cryptography to distribute the CA signing key would prevent one (or several) compromised node(s) from signing messages on behalf of the distributed CA.

In a related result, Kong, et al. [11, 12, 13] developed an interesting Threshold-RSA (TS-RSA) scheme specifically geared for MANET-s. Unfortunately, as pointed out in [4], TS-RSA is not verifiable, i.e., a new member cannot verify the correctness of the partial secret shares received from sponsors. Thus, malicious insiders cannot be detected.

An alternative Threshold-DSA (TS-DSA) scheme is proposed by Narasimha, et al [4]. This scheme provides verifiability and, hence, tolerates malicious insiders. However, TS-DSA only remains secure as long as there are less than $\lfloor \frac{t+1}{2} \rfloor$ malicious members (where t is the desired group threshold). Thus, it provides weaker security than a general threshold cryptosystem which is $(t-1)$-secure. Moreover, its efficiency is an issue of some concern [5].

In all of the above signature-based schemes, a joining node receives a certificate as its proof-of-membership. These approaches are interesting but all have some limitations. First, even state-of-the-art threshold signature schemes are too computationally expensive for MANET environments [5]. Second, in many applications, such as secure route discovery or secure routing, a node needs to share a secret key with a set of intermediate nodes on the path to the destination. Since the aforementioned approaches only provide certificates, the source has to perform a key exchange protocol with each intermediate node prior to communication. This is very expensive and impractical. In contract, our scheme uses threshold secret sharing technique. Furthermore, a node receives a set of private keys that it can use to locally compute (i.e., without communication) a secret with any other node in the network.

Recently, Zhu, et al. proposed a pair-wise key distribution scheme based on the combination of probabilistic key sharing and threshold secret sharing [14]. However, it is assumed that the nodes are *pre-configured* with some secrets before deployment which is not realistic in a typical MANET environment. Furthermore, two nodes need to communicate over several distinct paths to establish a shared key. In contrast, we do not assume any such pre-configuration and do not require nodes to communicate when establishing a secret key.

Ćapkun, et al. proposed a security association establishment protocol that makes use of the mobility of users [15]. Two nodes establish a security association when they are near each other, by using secure channels. As a node moves around, it establishes more and more security associations. When a node needs to establish a secret with another node, there are two possibilities: (1) they already have a security association, or (2) they have no security association and must use the help of "friends" to establish one. Despite the simplicity and elegance of this approach, it is mainly geared for highly mobile MANET-s. Furthermore, key derivation among two nodes that do not have a prior security association requires some communication, which is not always practical or even possible.

3 Building Blocks

This section describes the main techniques used in our proposal, namely the Blom key pre-distribution, Shamir secret sharing scheme, and verifiable secret sharing.

3.1 Notation

The notation used in this paper is summarized below:

M_i	member i.e., network node i
AN_i	authorization node i
t	node admission threshold
λ	number of private keys that M_i must store
N	maximum size of network nodes
n	number of founding members
FM	founding member
NID	network identity
ID_i	crypto-based identifier of M_i or AN_i
PK_i	public key of M_i
K_{ij}	secret key shared between M_i and M_j
$r_i(A)$	row of matrix A for M_i
$ss_i(x)$	secret share of value x for M_i
$pss_j^i(x)$	partial secret share of x for M_i by M_j
SL_i	sponsor list for M_i to reconstruct a secret
$H(x)$	hash value on input x
$E_k(x)$	encryption with a key k on input x
$MAC(k,x)$	message authentication code with key k on input x

3.2 Blom Key Pre-distribution

Blom proposed a key predistribution scheme that allows any pair of users in a group to compute a pairwise key without communicating [7]. This scheme is secure unless λ users collude (the parameter λ will be defined later). If less than λ users collude, then it is proven that the system is completely secure i.e., the colluding nodes can not compute any pair-wise keys other than their own. However, if λ or more users collude, the whole group is compromised and the colluding users can compute the pair-wise keys of all other members.

In Blom's proposal, a trusted dealer TD computes a $\lambda \times N$ matrix B over $\mathbb{Z}_q$, where N is the maximum size of the group, q is a prime, and $q > N$.

One example of such a matrix is a Vandermonde matrix whose element $b_{ij} = (g^j)^i \pmod q$ as seen below, where g is the primitive element of $\mathbb{Z}_q^*$.

$$B = \begin{bmatrix} b_{ij} = (g^j)^i \\ \text{for } i,j = 1, \cdots, \lambda \end{bmatrix} \pmod q$$

Note that this construction requires that $N\lambda < \phi(q)$ i.e., $N\lambda < q - 1$.

Since B is a Vandermonde matrix, it can be shown that any λ columns are linearly independent when $g, g^2, g^3, ..., g^N$ are all distinct [16]. The TD then creates a random $\lambda \times \lambda$ symmetric matrix D over $\mathbb{Z}_q$, and computes an $N \times \lambda$ matrix $A = (DB)^T$, where T indicates a transposition of the matrix.

The matrix B is published while the matrix D is kept secret by the TD. Since D is symmetric, if we define $K = AB$, we have:

$$K = (DB)^T B = B^T D^T B = B^T D B = (AB)^T = K^T.$$

This shows that K is also a symmetric matrix.

We assume that each user, M_i, is defined by an identifier, ID_i, such that $0 < i < N$. The TD then sends, over a secret channel, to each user M_i, the i^{th} row of the matrix A, denoted as $r_i(A)$, i.e., $r_i(A) = [a_{ij}]$ for $j = 1, ..., \lambda$.

A user M_i can then compute its key with user M_j as follows: $[K_{ij} = \sum_{\beta=1}^{\lambda} a_{i\beta} \cdot b_{\beta j}]$, where $b_{\beta j}$ is the element of B at row β and column j.

This key can be computed without communication since $b_{\beta j} = (g^j)^\beta \pmod q$. Similarly, user M_j can then compute its key with user M_i as follows: $[K_{ji} = \sum_{\beta=1}^{\lambda} a_{j\beta} \cdot b_{\beta i}]$. Since K is symmetric we have $K_{ij} = K_{ji}$, i.e., users M_i and M_j share a secret key. Note that each node does not have to store the whole matrix B only if he knows the public parameter g.

Since each pairwise key is represented by an element in $\mathbb{Z}_q$, q must be selected as the smallest prime number larger than 2^l, where l is the size in bits of the pairwise keys, for example 64.

3.3 Threshold Secret Sharing

The field of threshold cryptography [17] is concerned with distributing the ability to provide cryptographic services for reasons as improved fault tolerance (even though some nodes are unavailable, the others can still perform the task) and security (no single entity is entrusted to perform the task in its entirety). Consequently, it seems an ideal choice for providing various security services (such as authentication services and access control mechanisms) in MANET-s. Specifically, a (t, n) threshold cryptographic scheme allows n parties to share the ability to perform a cryptographic operation 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.

We use Shamir's secret sharing scheme [9] which is based on polynomial interpolation. To distribute shares among n users, a trusted dealer TD chooses a large prime q, and selects a polynomial $f(x)$ over $\mathbb{Z}_q$ of degree $t-1$ such that $f(0) = S$, where S is the group secret. The TD computes each user's share ss_i such that $ss_i = f(i) \pmod q$, 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: $f(0) = \sum_{i=1}^{t} ss_i \; l_i(0) \pmod q$, where $l_i(0) = \prod_{j=1, j\neq i}^{t} \frac{j}{j-i} \pmod q$.

Thus, S can be recovered only if at least t shares are combined. In other words, no coalition of $t-1$ members or fewer yields any information about S. The $l_i(0)$ are non-secret constants, and may be precomputed.

3.4 Verifiable Secret Sharing

If some nodes can become malicious or somehow compromised, they may attempt to "cheat" by using incorrect secret shares in order to deny service or simply disrupt the network. To deal with malicious insiders, a more advanced technique, *Verifiable Secret Sharing* (VSS) [18] can be used. It basically provides a means to detect incorrect secret shares. To be more specific, VSS requires a generator g of the group $\mathbb{Z}_p^*$ which has order q, s.t. $q|p-1$.

The procedure for the TD to distribute the shares is the same as in Section 3.3. VSS is achieved as follows. The TD randomly selects a polynomial $f(x)$ over $\mathbb{Z}_q$ such that $f(x) = \delta^{(0)} + \delta^{(1)}x + \cdots + \delta^{(t-1)}x^{t-1} \pmod q$, computes secret shares ss_i, and transfers them to each user securely. Also, the TD computes the *witness* W_α such that $W_\alpha = g^{\delta^{(\alpha)}} \pmod p$ for $\alpha = 0, \cdots, t-1$. Then, the TD publishes these W_α-s in some public domain (e.g., a directory server)[1]. When t members receive their share ss_i, each member M_i verifies ss_i by checking that $g^{ss_i} = \prod_{\beta=0}^{t}(W_\beta)^{\nu^\beta} \pmod p$.

4 TKP: Threshold Key Pre-distribution

We propose a solution to the key establishment problem that is based on a novel combination of the Blom key predistribution [7] and Shamir threshold secret sharing schemes [9]. We refer to this new scheme as *TKP (Threshold Key Pre-distribution)*.

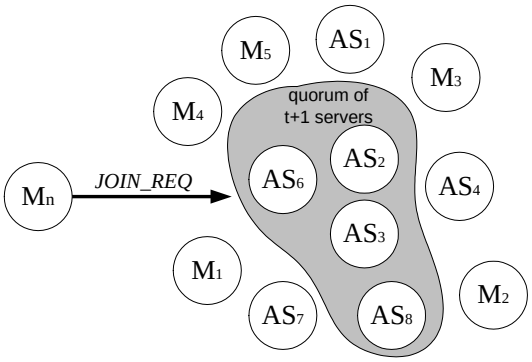


Figure 1: TKP Network Model. M_i and AN_i indicate a *member* and an *authorization node*, respectively.

4.1 Overview

Our proposal follows the model defined by Zhou and Haas [10] that suggests to distribute the CA over several nodes of the network and to use threshold cryptography to improve security and robustness.

In this paper, we use the following terminology (see Figure 1):

1. A *member* is a node that is part of the network. A member is also part of the routing infrastructure i.e., it routes packets for other members.

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

2. An *authorization node* is a member that can participate in the admission of a candidate member. A node needs to receive one partial secret token from at least t different authorization nodes to become a member. Similarly, a member can become an authorization node if it is approved by t current authorization nodes.

We also assume that each node is identified by a publicly known identifier, ID_i, and that the network is also identified by a publicly known identifier, NID.

The different steps of our proposal can be summarized as follows (this description has been intentionally simplified for clarity. A more precise description of each of these steps is presented in the following section):

1. *Bootstrapping:* A network is bootstrapped by either one single founding member or a set of founding members. The founding members compute the matrices D, B and A, defined in Section 3.2. The founding members then split the matrix D into n shares, $ss_i(D)$, such that at least t shares are required to reconstruct it. n authorization nodes are then selected and each of them receives one share of D.

2. *Member Admission:* A prospective member M_η initiates the protocol by sending a `JOIN_REQ` message to the network. An authorization node, that receives this `JOIN_REQ` message and approves the admission of M_η, replies, over a secure channel, with the row η of its share of matrix A (we explain in the following section how a share of the matrix A can be computed from a share of the matrix D). Once M_η received one share from at least t different authorization nodes, it can retrieve the row η of matrix A using Lagrange interpolation. It can then use these system secrets to compute a key with any other member of the network according the Blom key establishment protocol described in Section 3.2.

   As explained in Section 4.2, M_η must verify the validity of the reconstructed secrets to protect against Denial of Service (DoS) attack.

3. *Authorization Node Admission:* A member M_η can be promoted to become an authorization node by receiving *partial* shares of the whole matrix D from t current authorization nodes. It can then use these shares to reconstruct its *share* of the matrix A, and consequently use it to admit new members. Also, as in the member admission phase, the reconstructed share of D need to be verified.

4. *Secret Key Computation:* The pair-wise key computation procedure is the same as the one described in Section 3.2.

Note that our scheme is completely distributed and as such can be qualified as a peer-to-peer scheme; nodes get admitted and promoted to become authorization nodes by a quorum

of their peers. A network can get bootstrapped by a set of nodes that get together and compute the security parameters of the network in a distributed way. Our scheme does not require any kind of central authority or trusted third party.

4.2 Verifiability and Traceability

A malicious node can easily launch a DoS attack toward a candidate node by inserting incorrect secret shares. This attack would actually deny or disrupt the service to legitimate nodes. To deal with this important problem a node must be able to verify the validity of its reconstructed secrets (i.e., its row of the matrix A or its share of the whole matrix D) before using them. This is what we call *verifiability* in the rest of the paper.

Also, when the node detects that its secrets are not valid, it must be able to trace the bogus shares in order to replace them and/or revoke the malicious participants. This functionality is provided by the *traceability* procedures. Note that verifying the shares' origin, for example via signatures, is not enough to provide traceability since it does not protect against compromised nodes that would signed correctly but send bogus shares. Instead, *traceability* must allow to verify the validity of the shares themselves.

Note that *verifiability* is always required. *Traceability* is only necessary when a node detects (from the verifiability service) that its reconstructed secrets are not valid.

5 Protocol Description

This section details the different steps of our protocol.

5.1 Bootstrapping

In our scheme, a network can be bootstrapped (i.e., initialized) by one node (centralized bootstrapping) or a set of t or more nodes (distributed bootstrapping). In the former case, one node (the founding member) generates all the necessary network and security parameters. He then configures some nodes as authorization nodes. In the latter case, the parameters are jointly generated by a set of t or more founding members conjointly. Each founding member contributes to these parameters without learning the final value of the matrix D. Note that the founding members do not have to remain members of the network for its whole lifetime. They can initiate the networks and then leave it.

5.1.1 Centralized Bootstrapping

The centralized bootstrapping proceeds as follows. First, a founding member FM generates the network parameters, namely n, N, λ, t, q, p, g, and the matrices $D = [d_{ij}]$ and $B = [b_{ij}]$, where n is the number of authorization nodes, N is the maximum numbers of nodes in the network, (λ, q, p, g) are the security parameters described in Section 3, B is the public matrix, and D is the matrix of secrets.

Next, the FM publishes $(n, N, \lambda, t, q, p, g, B)$ in some public directory, but keeps D secret. It then computes the matrix A and must send a share of the whole matrix A to each of n authorization nodes. However, the size of the matrix A is $N \times \lambda$. If N is large (we will see in Section 6.3 that $N = 2^{64}$), the bandwidth cost can be extremely high. Fortunately, as shown below, an authorization node AN can reconstruct its share of the matrix A from its share of the matrix D. As a result, only a share of the matrix D has to be sent to each AN. This reduces the bandwidth cost significantly. Therefore, the goal of bootstrapping is to configure each authorization node AN_v $(v \in \{1, n\})$ with a share of matrix D, denoted by $ss_v(D)$, and with its own row of matrix A, denoted by $r_v(A)$.

To compute $ss_v(D)$, FM selects polynomials for each element d_{ij} of $\lambda \times \lambda$ matrix D. Each polynomial is defined as follows; $f_{d_{ij}}(x) = \sum_{\alpha=0}^{t-1} \delta_{ij}^{(\alpha)} \cdot x^{\alpha} \pmod q$ such that $\delta_{ij}^{(0)} = d_{ij}$. The share of matrix D is made up of shares of its elements $ss_v(d_{ij})$. In other words, $ss_v(D) = [ss_v(d_{ij})] = [f_{d_{ij}}(v)]$ for $i, j = 1, \cdots, \lambda$.

As for $r_v(A)$ such that $r_v(A) = [a_{vj}]$ for $j = 1, \cdots, \lambda$, each element of $r_v(A)$ is simply computed by FM since it knows the secret matrix D. That is, $a_{vj} = \sum_{\beta=1}^{\lambda} d_{j\beta} \cdot b_{\beta v} \pmod q$. Then FM distributes $ss_v(D)$ and $r_v(A)$ to each AN_v.

In addition, FM computes three different types of VSS witnesses (which will be used in the traceability procedures defined in Section 6.2), $W_{ij}^{(\alpha)}$, $W_{C_j}^{(\alpha)}$, and $W_D^{(\alpha)}$ as follows: $W_{ij}^{(\alpha)} = g^{\delta_{ij}^{(\alpha)}} \pmod p$, $W_{C_j}^{(\alpha)} = g^{\sum_{i=1}^{\lambda} \delta_{ij}^{(\alpha)}} \pmod p$, and $W_D^{(\alpha)} = g^{\sum_{i=1}^{\lambda} \sum_{j=1}^{\lambda} \delta_{ij}^{(\alpha)}} \pmod p$ for $i, j = 1, \cdots, \lambda, \alpha = 0, \cdots, t-1$. These witnesses are called *element witness*, *column witness*, and *matrix witness* respectively.

5.1.2 Distributed Bootstrapping

A network can alternatively be bootstrapped by a set of t or more founding members. Due to Pedersen [19], the secret matrix D can be generated in fully distributed manner. In this proposal, a group of members (the founding members in our scenario) collectively compute shares corresponding to Shamir secret sharing of a random value without a centralized trusted dealer. The main idea is that the polynomial itself is shared such that $f_{d_{ij}}(x) = \sum_{v=1}^{n} f_{d_{ij}}^{v}(x) \pmod q$, where $f_{d_{ij}}^{v}(x)$ is the polynomial of founding member M_v over $\mathbb{Z}_q$. In other words, each founding member selects its own polynomial. The full polynomial is jointly constructed by summing up all of these partial polynomials. It is so-called *Joint Secret Sharing (JSS)*. For more details, refer to [19].

Once the matrix D is securely and collectively generated, the rest of procedure for bootstrapping is same as in centralized one.

Note that in the centralized mode, the FM is similar to a trusted third party and is, therefore, a single point of failure. However, the FM is only required during the bootstrapping phase and can disappear once the authorization nodes have been configured. The distributed bootstrapping mode provides a more secure solution since the security parame-

ters are defined by a group of nodes and no node has a complete knowledge of the generated secret matrix D.

5.2 Member Admission

Let n be the number of current authorization nodes. In order to join the network, a prospective node M_η must collect at least t shares of matrix A's row η from current authorization nodes. Figure 2 shows the protocol message flow for this admission process [2]. The goal is for M_η to obtain his private matrix row $r_\eta(A)$ which can then be used to compute a secret key with any other node of the network.

$msg1(M_\eta \rightarrow AN_\nu)$:	$REQ = \{ID_\eta, y_\eta, PK_\eta\}$, $S_\eta(REQ)$	(1)
$msg2(M_\eta \leftarrow AN_\nu)$:	$REP = \{ID_\nu, y_\nu, PK_\nu\}$, $S_\nu(REP, H(REQ))$	(2)
$msg3(M_\eta \rightarrow AN_\mu)$:	$MAC(DHK_{\eta\mu}$, $H(msg1, msg2))$	(3)
$msg4(M_\eta \leftarrow AN_\mu)$:	$E_{DHK_{\eta\mu}}\{ss_\mu(r_\eta(A))\}$	(4)

Figure 2: Member Admission Protocol. The messages in step (1) and (2) are sent to/from all AN_ν-s ($1 \leq \nu \leq n$). The messages in step (3) and (4) are communicated with only t out of n authorization nodes, i.e., AN_μ ($\mu \in_R \nu, |\mu| = t$).

1. M_η sends to at least t current authorization nodes AN_ν-s ($\nu \in \{1, n\}$) a signed $JOIN\_REQ$ message which contains his identity ID_η, his public DH component $y_\eta (= g^{x_\eta} \mod p)$, and his public key PK_η. Note that PK_η is required to verify the signature and the validity of ID_η. The details about how PK_η and ID_η are generated and verified are discussed in Section 6.3.

2. After verifying the signed $JOIN\_REQ$, the AN-s who wish to participate in the admission process of M_η reply with a signed message containing their respective values ID_ν, y_ν, and PK_ν.

3. M_η selects t sponsors AN_μ, computes a secret key $DHK_{\eta\mu}$ with each of them (using the DH key exchange protocol - see Section 6.3 for more details) and replies with an authenticated acknowledgment message to each of them.

4. Each sponsoring node (AN_μ) receiving $msg3$, computes the secret key $DHK_{\eta\mu}$ and replies with row η of it share of the matrix A, $ss_\mu(r_\eta(A))$. The elements of $ss_\mu(r_\eta(A))$ are computed as $ss_\mu(r_\eta(a_{\eta j})) = \sum_{\beta=1}^{\lambda} ss_\mu(d_{j\beta}) \cdot b_{\beta\eta} \pmod q$, for $j = 1, \cdots, \lambda$. This message is encrypted with $DHK_{\eta\mu}$.

[2] In order to secure the protocol against common *replay* attacks [6], we note that it is necessary to include timestamps, nonces and protocol message identifiers. However, in order to keep our description simple, we omit these values.

5. M_η decrypts the messages it receives from the different AN-s and calculates his own $r_\eta(A)$ by adding up all $ss_\mu(r_\eta(A))$-s as follows:

$$r_\eta(A) = \sum_{\mu=1}^{t} ss_\mu(r_\eta(A)) \cdot l_\mu(0) \pmod q$$
$$= [\sum_{\mu=1}^{t} ss_\mu(r_\eta(a_{\eta j})) \cdot l_\mu(0)]$$
$$\text{for } j = 1, \cdots, \lambda.$$

5.3 Authorization Node Admission

A member M_η that becomes an authorization node (AN_η) needs to receive a valid share of the whole matrix D. Figure 3 shows the protocol message flow for the anthorization node admission process.

$msg1(M_\eta \rightarrow AN_\nu)$:	$REQ = \{ID_\eta, y_\eta, PK_\eta\}$, $S_\eta(REQ)$	(1)
$msg2(M_\eta \leftarrow AN_\nu)$:	$REP = \{ID_\nu, y_\nu, PK_\nu\}$, $S_\nu(REP, H(REQ))$	(2)
$msg3(M_\eta \rightarrow AN_\mu)$:	$SL_\eta, MAC(DHK_{\eta\mu}, H(SL_\eta, msg1, msg2))$	(3)
$msg4(M_\eta \leftarrow AN_\mu)$:	$E_{DHK_{\eta\mu}}\{pss_\mu(r_\eta(D))\}$	(4)

Figure 3: Authorization Node Admission Protocol. The messages in step (3) and (4) are communicated with only t out of n authorization nodes, i.e., AN_μ ($\mu \in_R \nu, |\mu| = t$).

1. Same as the step (1) in Section 5.2.
2. Same as the step (2) in Section 5.2.
3. M_η selects t sponsors $AN_\mu(\mu \in_R \nu, |\mu| = t)$, computes a secret key $DHK_{\eta\mu}$ with each of them as in a member admission process, forms a sponsor list SL_η which contains the ID-s of the t selected sponsors, and replies with message $msg3$ to each of the sponsors.
4. Each sponsoring node (AN_μ) computes the secret key $DHK_{\eta\mu}$ and replies with the *shuffled* partial share of matrix D, $pss^\eta_\mu(D)$, such that $pss^\eta_\mu(D) = [pss^\eta_\mu(d_{ij})] = [ss_\mu(d_{ij}) \cdot l_\mu(\eta)] \pmod q$ for $i, j = 1, \cdots, \lambda$. This message is encrypted using $K_{\eta\mu}$.

   Note that the Lagrange coefficients $l_\mu(\eta)$ are publicly known, and therefore, M_η can derive $ss_\mu(d_{ij})$ from $pss^\eta_\mu(d_{ij})$. This can be prevented using the *shuffling* technique proposed in [11] by adding extra random value R_{ij} to each share. These R_{ij}-s are secret values and must sum up to zero by construction. They must be securely shared among the t sponsoring AN-s.

5. Finally, M_η calculates his own share of the matrix D, $ss_\eta(D)$, by adding up the values obtained in the last step such that $ss_\eta(D) = \sum_{\mu=1}^{t} pss_\mu^\eta(D) = [\sum_{\mu=1}^{t} pss_\mu^\eta(d_{ij})] \pmod q$ for $i, j = 1, \cdots, \lambda$.

5.4 Secret Key Computation

When a node, M_i, reconstructs its private row of matrix A, $r_i(A) = [a_{i1}, \cdots, a_{i\lambda}]$, he can compute a secret key, K_{ij}, with any other node, M_j, of the network as follows:

Since $a_{ij} = \sum_{\alpha=1}^{\lambda} d_{j\alpha} \cdot b_{\alpha i}$ and $d_{ij} = d_{ji}$,

$$\begin{aligned} K_{ij} &= \sum_{\beta=1}^{\lambda} a_{i\beta} \cdot b_{\beta j} \\ &= \sum_{\beta=1}^{\lambda} \sum_{\alpha=1}^{\lambda} d_{\beta\alpha} \cdot b_{\alpha i} \cdot b_{\beta j} \\ &= \sum_{\alpha=1}^{\lambda} \sum_{\beta=1}^{\lambda} d_{\alpha\beta} \cdot b_{\beta j} \cdot b_{\alpha i} \\ &= \sum_{\alpha=1}^{\lambda} a_{j\alpha} \cdot b_{\alpha i} \\ &= K_{ji} \pmod q. \end{aligned}$$

Note that these keys do not have to be computed in advance but can be computed *on-the-fly*.

6 Security Considerations

This section presents the *verifiability* and *traceability* procedures that we developed to secure our scheme against the DoS attacks discussed in Section 4.2.

6.1 Verifiability

At the end of the admission protocols for a member or an authorization node, M_η obtains his own $r_\eta(A)$ or $ss_\eta(D)$ from the quorum of t authorization nodes, respectively. Before using $r_\eta(A)$ or $ss_\eta(D)$ for key computation or future admission, a node must verify if they are correctly computed since there might be a malicious responder who participated in this admission process. We therefore propose two verification mechanisms called *member verifiability* and *authorization node verifiability*, respectively.

6.1.1 Member Verifiability

The VSS technique presented in Section 3.4 will be a useful tool for the verifiability. However, we claim that the verifiability must be a very inexpensive operation since it will be performed frequently (whenever a node joins a network). The proposed mechanism to verify the validity of $r_\eta(A)$ is as follows.

1. *When an AN_μ sends the shares to the node M_η it also sends a well-known message, such as "Welcome to network NID" encrypted with the pair-wise key shared between AN_μ and M_η (since the AN knows the node identifier ID_η, it can compute the pair-wise key $K_{\mu\eta}$). This will be part of step (4) in Fig. 2.*

2. *After M_η reconstructs its systems secrets $r_\eta(A)$, it can then try to decrypt one of the welcome messages received from the authorization nodes and verify whether $r_\eta(A)$ is correctly computed.*

6.1.2 Authorization Node Verifiability

Additionally, if M_η reconstructs a secret share $ss_\eta(D)$ to become an authorization node, it must verify its correctness, too. If $ss_\eta(D)$ is correct, it can be used for future admission of other nodes. One easy way for M_η to verify the validity of $ss_\eta(D)$ is to try to use it to reconstruct its row of the matrix A (i.e., $r_\eta(A)$). Let us say that $r_\eta(A)$ was computed from the shares $ss_a(r_\eta(A))$, $ss_b(r_\eta(A))$, $ss_b(r_\eta(A))$ that it received from AN_a, AN_b and AN_c (for $t = 3$). M_η can then compute $r'_\eta(A)$ from the shares $ss_a(r_\eta(A))$, $ss_b(r_\eta(A))$ and $ss_\eta(r_\eta(A))$ (that can easily be computed from $ss_\eta(D)$). If $r'_\eta(A)$ is equal to $r_\eta(A)$ then the share $ss_\eta(D)$ is correct - otherwise it must be rejected.

6.2 Traceability

The verifiability procedures previously described allow a node to verify the validity of the secret (which is either a row of the matrix A or a share of the whole matrix D) that it reconstructed from t shares. However, the above procedure cannot be used to identify the bogus shares, in case the verification procedure fails (i.e., detects that the reconstructed secrets are invalid).

In this section, we present two different traceability procedures. The first – *external attack traceability*– traces external malicious nodes, i.e., malicious nodes that are not part of the MANET and just try to attack the network by sending bogus shares to new member or AN candidates. The second – *internal attack traceability* – is useful to detect attacks coming from current legitimate AN-s that either turned bad or got compromised.

Both procedures uses the previously described VSS technique in some innovative ways. Since the internal traceability procedure is quite costly, it is recommended to use the external attack traceability first and use the internal attack traceability procedure only if the

first one was unsuccessful.

6.2.1 External Attack Traceability

With this procedure, a node M_η, instead of verifying individual element of a share (row or matrix), verifies the sum of the elements of the share. As a result, instead of applying the VSS technique λ or λ^2 times, we only apply it once. This, of course, improves performance considerably.

More specifically, M_η verifies the validity of the share $ss_\mu(r_\eta(A))$ or $pss^\eta_\mu(D)$ using the VSS technique together with the *column witnesses* or *matrix witnesses*, defined in Section 5.1.1, as follows:

M_η first computes $\sigma_{ss_\mu(r_\eta(A))}$ by summing up all elements of $ss_\mu(r_\eta(A))$; i.e., $\sigma_{ss_\mu(r_\eta(A))} = \sum_{j=1}^{\lambda} ss_\mu(a_{\eta j}) \pmod q$. The validity of $\sigma_{ss_\mu(r_\eta(A))}$ can be verified by checking the following equality:

$$g^{\sigma_{ss_\mu(r_\eta(A))}} \stackrel{?}{=} \prod_{\alpha=0}^{t-1}\prod_{k=1}^{\lambda}\left(W_{C_k}^{(\alpha)}\right)^{\mu^\alpha \cdot b_{k\eta}} \pmod p$$

Similarly, a node can verify the validity of a partial of the matrix D, $pss^\eta_\mu(D)$, after computing $\sigma_{pss^\eta_\mu(D)} = \sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} pss^\eta_\mu(d_{ij}) \pmod q$, as follows:

$$g^{\sigma_{pss^\eta_\mu(D)}} \stackrel{?}{=} \prod_{\alpha=0}^{t-1}\left(W_D^{(\alpha)}\right)^{\mu^\alpha \cdot l_\mu(\eta)} \pmod p$$

proof: Since $ss_\mu(a_{\eta j}) = \sum_{k=1}^{\lambda} ss_\mu(d_{jk}) \cdot b_{k\eta}$, $ss_\mu(d_{jk}) = f_{d_{jk}}(\mu) = \sum_{\alpha=0}^{t-1} \delta_{jk}^{(\alpha)} \cdot \mu^\alpha \pmod q$, and $W_{C_j}^{(\alpha)} = g^{\sum_{i=1}^{\lambda} \delta_{ij}^{(\alpha)}}$ for $\alpha = 0, \cdots, t-1$,

$$\begin{aligned}
g^{\sigma_{ss_\mu(r_\eta(A))}} &= \prod_{\alpha=0}^{t-1}\prod_{k=1}^{\lambda}\left(W_{C_k}^{(\alpha)}\right)^{\mu^\alpha \cdot b_{k\eta}} \\
&= \prod_{\alpha=0}^{t-1}\prod_{k=1}^{\lambda}\left(\prod_{j=1}^{\lambda} g^{\delta_{jk}^{(\alpha)}}\right)^{\mu^\alpha \cdot b_{k\eta}} \\
&= g^{\sum_{j=1}^{\lambda}\sum_{k=1}^{\lambda}\left(\sum_{\alpha=0}^{t-1}\delta_{jk}^{(\alpha)}\cdot\mu^\alpha\right)\cdot b_{k\eta}} \\
&= g^{\sum_{j=1}^{\lambda}\sum_{k=1}^{\lambda} ss_\mu(d_{jk})\cdot b_{k\eta}} \\
&= g^{\sum_{j=1}^{\lambda} ss_\mu(a_{\eta j})} \pmod p.
\end{aligned}$$

Also, since $W_D^{(\alpha)} = g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda}\delta_{ij}^{(\alpha)}} \pmod p$,

$$
\begin{aligned}
g^{\sigma_{pss_\mu^\eta(D)}} &= \prod_{\alpha=0}^{t-1} \left(W_D^{(\alpha)}\right)^{\mu^\alpha \cdot l_\mu(\eta)} \\
&= \prod_{\alpha=0}^{t-1} \left(g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} \delta_{ij}{}^{(\alpha)}}\right)^{\mu^\alpha \cdot l_\mu(\eta)} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda}\left(\sum_{\alpha=0}^{t-1} \delta_{ij}{}^{(\alpha)} \cdot \mu^\alpha\right) \cdot l_\mu(\eta)} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} ss_\mu(d_{ij}) l_\mu(\eta)} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} pss_\mu^\eta(d_{ij})} \pmod p.
\end{aligned}
$$

If the above verification fails, M_η concludes that M_μ is cheating. Otherwise, the malicious node is a group member and thus the following procedure must be used.

6.2.2 Internal Attack Traceability

If a malicious insider (AN_m), who has valid $ss_m(D)$, modifies a value in $ss_m(D)$ so that the sum of $ss_m(D)$ remains unchanged (say, $ss_m^{'}(D)$ such that $\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} ss_m^{'}(d_{ij}) = \sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} ss_m(d_{ij})$), the *external attack traceability* procedure does not work. In order to protect against such an internal attack, VSS must be applied individually to each of elements of $pss_\mu(r_\eta(A))$ and/or $ss_\mu^\eta(D)$. In other words, *internal member traceability* and *internal authorization node traceability* are provided by checking that $g^{ss_\mu(a_{\eta j})} = \prod_{\alpha=0}^{t-1}\prod_{k=1}^{\lambda}(W_{jk}^{(\alpha)})^{\mu^\alpha \cdot b_{k\eta}} \pmod p$ and $g^{pss_\mu^\eta(d_{ij})} = \prod_{\alpha=0}^{t-1}(W_{ij}^{(\alpha)})^{\mu^\alpha \cdot l_\mu(\eta)} \pmod p$, where $i, j = 1, \cdots, \lambda$.

Obviously, these tracing mechanisms for an internal attack are more expensive than the external ones. However, we argue that internal attacks are more difficult to perform than external attacks (since it requires compromised members) and should, hopefully, be less frequent.

6.3 Discussions

6.3.1 Identifier Configuration

In our scheme, the identifier ID_i of each node M_i must be *unique* and *verifiable*.

- It must be unique because if two nodes get the same ID_i, they will receive the same row of the matrix A.
- It must be verifiable, i.e., a node must be able to prove ownership of its identifier, otherwise a malicious node could use someone-else ID and get its secret from the authorization nodes. This attack could be used to impersonate a node or eavesdrop on its traffic.

One simple solution is that once an AN assigns an ID to a node it advertises it to all other $n-1$ AN-s. This solution is not scalable and not robust against network partition. In fact if the network partitions into two networks, each partition could assign the same index to different nodes. We instead propose a solution based on *Crypto-Based ID (CBID)* [20]: The ID_i is chosen by the node itself from an ephemeral public/private key pair. More specifically, the node computes ID_i as follows: $ID_i = H_{64}(PK_i|NID)$, where PK_i is M_i's temporary public key, NID is the network identifier and $H_{64}(\cdot)$ a 64-bit long hash function. When a node contacts an AN, it sends its identifier ID_i together with its ephemeral public key PK_i and signs a challenge sent by the AN. Upon reception of the signature, the AN can verify that the ID_i actually belongs to the node (by verifying the signature and that the ID_i was generated as $H_{64}(PK_i|NID)$. Note that the PK_i does not need to be certified and therefore no PKI is required. The identifier is *verifiable* because a node that does not know the private key, associated with the public key used to generate an ID, can not claim to own it. Furthermore since i is computed from a hash function, collision probability between two nodes is very low. As a result, the identifier are *statistically unique*. Note that this solution requires that $N = 2^{64}$. However, as we will see this has no effect on the performance or scalability of our proposal.

6.3.2 Secure Channel Establishment

In the proposed protocols, the channels between a node and its authorization nodes must be authenticated and encrypted. It has to be authenticated because the AN must be sure that it is sending the shares to the correct node (i.e., the node that claims to own the identifier). Otherwise, the AN-s could send the shares to an impersonating node. It has to be private because otherwise a malicious node that eavesdrops on the shares sent to a node could reconstruct the node's secret and impersonate it.

Establishing an authenticated and private channel usually requires the use of certificates, which bind identities to public keys, and an access to a PKI. However, PKI are not always available in MANET environments. Fortunately in our case, what is really needed is a way to bind an identifier to a public key, where the identifier is a number that identifies one row of the matrix A. This binding is actually provided by *CBID*, described previously. As a result, certificates and PKI are not required. Therefore, the PK-s that are sent in message 1 and 2 of the protocols described in Sections 5.2 and 5.3 do not need to be certified. To establish a secure channel, a member and an AN use the regular authenticated DH key exchange protocol. Each node verifies that the PK it received from its peer is valid by verifying that the signature is correct and that the key was used to generate the peer ID.

6.3.3 Parameters selection

The security of our scheme relies on two security parameters t and λ which should be selected carefully. The system remains secure as long as less than t authorization nodes and less than

λ nodes (regular members or authorization nodes) get compromised. The values of t and λ must be selected by the founding members when they bootstrap the network. Of course larger values of t and λ increase security, since a malicious node has to compromise more nodes, but it also degrades the performance (since a node would have to collect a higher number of shares). For some networks, authorization nodes might be better protected and therefore more difficult to compromise than regular nodes. In this case, it is reasonable to use a value of t that is smaller than λ. If AN-s and nodes are equally protected then it is suggested to set $t = \lambda$. In any case, we must have $\lambda \geq t$.

7 Performance Evaluation

We implemented and evaluated the *TKP* protocol in a real MANET environment. This section presents the performance of each phase of our scheme. It also compared them with the performance of the previously proposed TS-DSA scheme [4, 5], wherever applicable.

7.1 Experimental Setup

We implemented the *TKP* protocol on top of the OpenSSL library [21]. It is written in C for Linux, and consists of about 10,000 lines of code.

We used two laptops and one PDA; a laptop with a P3-1.2GHz CPU and 512MB memory, a laptop with a P3-800MHz CPU and 384MB memory, and a PDA with an Intel Xcale 400MHz CPU and 64MB memory. Each device ran Linux 2.4 and was equipped with a 802.11*b* wireless card configured in ad-hoc mode.

In our experiments, each node was emulated by a daemon and each machine was running several daemons. This allowed us to emulate a large ad-hoc network with only three machines.

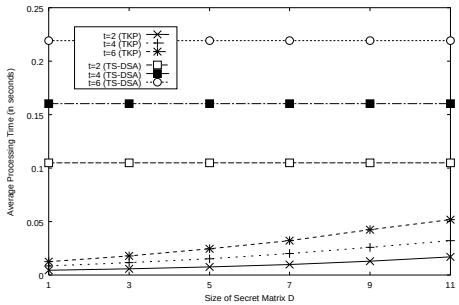
These experiments were performed with the different threshold values t=2, 4, and 6. All experiments were repeated 1,000 times for each measurement in order to get fairly accurate average results. The size of the parameters q and p was set to 160-bits and 1024-bits, respectively.

7.2 Admission Cost

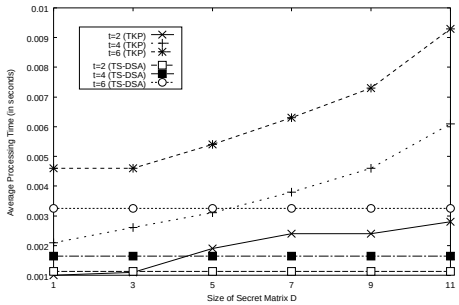
In these experiments, we measured the total processing time from the sending of the `JOIN_REQ` to the reception of (a) the new $r_\eta(A)$ and (b) $ss_\eta(D)$ later on. Thus, the result not only includes the average computation time of the basic operations, but also the communication costs such as packet encoding and decoding time, the network delay, and so on.

Figure 4 shows the average admission time for a member and an authorization node with varying λ for different values of the threshold t.

The cost for a new member to join the network with *TKP* is similar to in *TS-DSA*. In fact, both schemes have the same computation complexity: *TKP* takes $O(t)$ exponentia-



(a) Member



(b) Authorization Node

Figure 4: Admission Cost ($|p|$=1024). The X-axis indicates λ. The results with TKP are shown only when $\lambda \geq t$.

tions [3] to establish a secure channel and *TS-DSA* requires $O(t)$ exponentiations for signature generation and verification.

Similarly the *AN* admission costs of both schemes are similar: the reconstruction of secret share(s) only required $O(\lambda^2 t)$ modulus a 160-bit long integer for TKP and $O(t)$ *additions* modulus a 1024-bit long integer for $TS-DSA$, which are both negligible. The *AN* admission cost of both schemes is actually dominated by the cost DH key agreement protocol which requires $O(t)$ exponentiations.

7.3 Verifiability Cost

When a node joins a network, it must verify the validity of the reconstructed secrets, as described in Section 6.1. This section evaluates the cost of the verifiability procedures.

As shown in Figure 5, *TKP* outperforms *TS-DSA* for both member and authorization node verification. The *TKP* member verification procedure only requires $O(\lambda)$ multiplications modulus a 160-bit long integer

while *TS-DSA* requires three expensive exponentiations (one for signature generation and two for signature verification as in a standard DSA signature scheme). The $TS-DSA$ *AN* verification procedure requires $O(t)$ exponentiations for VSS, whereas the *TKP* procedure only requires $O(\lambda)$ multiplications modulus a 160-bit long integer.

7.4 Traceability Cost

As described in Section 6.2, the traceability procedures are used to identify the cheating or misbehaving nodes during the admission protocols. This section evaluates their performance.

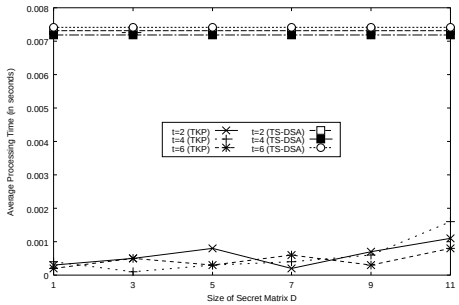
Figure 6 displays the cost of the external attack traceability procedures. The cost of the member traceability procedure with *TKP* depends on the value of λ as well as t. As a result, this cost increases when λ gets larger. The complexity of the TKP external node traceability procedure is $O(\lambda t)$ exponentiations, compared to $O(t)$ exponentiations for *TS-DSA*. The *AN* traceability procedures of *TKP* and *TS-DSA* have similar performance: both procedures require $O(t^2)$ exponentiations.

The internal attack traceability procedures are more expensive since they require $O(\lambda^2 t)$ (for member traceability) and $O(\lambda^2 t^2)$ (for AN traceability) exponentiations. However we expect these procedures to be executed very infrequently.

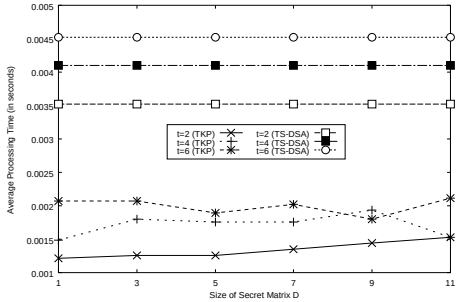
7.5 Key Computation Cost

Table 1 compares the cost of computing a pair-wise key in our scheme with the cost of a *DH* key derivation protocol. These experiment were performed on a 800MHz Laptop with $t = 4$. The results show that *TKP* performs significantly better than a *DH* key establishment protocol. The achieved gains range from 222 ($\lambda = 1$) to 59 ($\lambda = 9$). In other words, *TKP* is 59 to 222 times faster than *DH* to establish a shared secret key.

[3] All the exponentiations in the rest of the paper are modulus a 1024-bit long integer unless explicity specified.



(a) Member

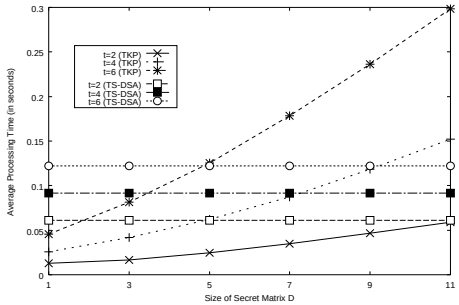


(b) Authorization Node

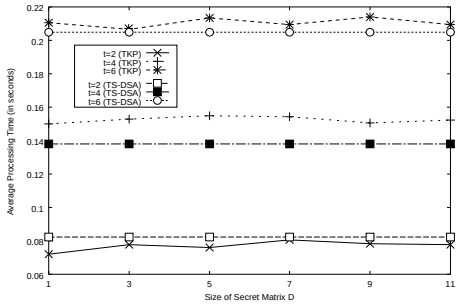
Figure 5: Verifiability Cost ($|p|$=1024). The graphs show the cost to verify the correctness of the newly generated (a) $r_\eta(A)$ and $ss_\eta(D)$.

Table 1: Key Computation Cost ($|p| = 1024$, $t = 4$, in msecs)

λ	TKP	DH	Gain
1	0.148	33.00	222
3	0.212	33.00	148
5	0.343	33.00	96
7	0.476	33.00	69
9	0.556	33.00	59



(a) Member



(b) Authorization Node

Figure 6: Traceability Cost with External Attack ($|p|$=1024). The graphs show the time that M_η takes to detect which malicious node sent a false value if any.

These results were actually expected because in *TKP* a pair-wise computation only requires λ modular *multiplications* where a modulus size is 160 bits. In contrast, a *DH* key derivation requires one expensive modular exponentiation with a modulus size of 1024 bits.

This is an important result and a major contribution of our scheme since key computation is a frequent operation in Ad-hoc routing protocols that gains to be optimized.

7.6 Comparison Summary

To conclude the discussion of the experiments, we now summarize and compare *TKP* with *TS-DSA* [5], [4]. Table 2 summarizes computation costs in terms of basic operations which mostly affects the performance of each scheme.

Table 2: Computation Comparison (No. of Expos)

		TKP	TS-DSA
Member	Admission	$O(t)$	$O(t)$
	Verifiability	-	$O(1)$
	Traceability	$O(\lambda t)$	$O(t)$
Auth. Node	Admission	$O(t)$	$O(t)$
	Verifiability	-	$O(t)$
	Traceability	$O(t^2)$	$O(t^2)$

Table 3: Bandwidth Comparison

	TKP	TS-DSA
Member	$O(\lambda t\|q\|)$	$O(t\|GMC\|)$
Auth. Node	$O(\lambda^2 t\|q\|$	$O(t\|q\|)$

Table 3 compares the respective bandwidth costs. The *TKP* member admission procedure requires less bandwidth than in *TS-DSA*, which uses GMC(Group Membership Certificate). Certificate size, CS, is relatively large, e.g., 5KB with 1024-bit DSA parameters. As a result, $(2t+1) * CS$ bytes must be transfered with *TS-DSA*, whereas, only $\lambda * t * 20$ bytes are needed in *TKP* since q is 160-bit integer.

8 Conclusions

In this paper, we presented a novel solution to the key predistribution problem in MANET. Our scheme, *TKP*, is based on the combination of two techniques: key predistribution and threshold secret sharing. It is secure against collusive attacks by $t-1$ compromised authorization nodes and $\lambda - 1$ regular nodes of the network where t is the number of shares a node must receive to become part of the network and λ is the number of private keys that each node must store.

The proposed protocol is robust against DoS attacks since a node can verify the validity of the shares it receives and possibly identify the incorrect ones.

The most commonly-used model in MANET key distribution consists of distributing the CA among several nodes and using threshold cryptography. When a node joins a network, it receives several partial signatures of its public key and reconstructs a valid certificate from them. When two nodes need to establish a secret, they execute a standard authenticated DH key exchange. In contrast, in our scheme, a node does not receive a certificate but a set of secrets. These secrets can be used to compute a key with any other node of the network and prove its membership.

We compared through experimentations the performance of the certificate-based *TS-DSA* scheme with the performance of our scheme. We compared the one-time membership cost for a member and an authorization node. Under normal circumstances, both schemes have similar performance. However, key derivation is 59 to 222 times more efficient with the proposed scheme than with the standard Diffie-Hellman approach. This is an important result since key computation is a frequent operation in Ad-hoc routing protocols that gains to be optimized. Our scheme is not only more practical (because pair-wise key can be computed locally without communication) but also provides much better performance.

References

[1] Yih-Chun Hu, Adrian Perrig, and David B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking* (MobiCom 2002).

[2] Yih-Chun Hu, David B. Johnson, and Adrian Perrig, "Sead: Secure efficient distance vector routing in mobile wireless ad hoc networks," in *Fourth IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA '02), June 2002, pp. 3–13.

[3] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," 2002.

[4] Maithili Narasimha, Gene Tsudik, and Jeong Hyun 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.

[5] Nitesh Saxena, Gene Tsudik, and Jeong Hyun 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.

[6] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of applied cryptography*, CRC Press series on discrete mathematics and its applications. 1997, ISBN 0-8493-8523-7.

[7] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," in *EUROCRYPT '84*. IACR, 1984, LNCS.

[8] T. Leighton and S. Micali, "Secret-key agreement without public-key cryptography," in *CRYPTO'93*, 1993.

[9] Adi Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[10] Lidong Zhou and Zygmunt J. Haas, "Securing Ad Hoc Networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24–30, 1999.

[11] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang, "Providing Robust and Ubiquitous Security Support for MANET," in *IEEE 9th International Conference on Network Protocols (ICNP)*, 2001.

[12] Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang, "Self-securing Ad Hoc Wireless Networks," in *Seventh IEEE Symposium on Computers and Communications (ISCC '02)*, 2002.

[13] Jiejun Kong, Haiyun Luo, Kaixin Xu, Daniel Lihui Gu, Mario Gerla, and Songwu Lu, "Adaptive Security for Multi-level Ad-hoc Networks," in *Journal of Wireless Communications and Mobile Computing (WCMC)*, 2002, vol. 2, pp. 533–547.

[14] Sencun Zhu, Shouhoui Xu, Sanjeev Setia, and Sushil Jajodia, "Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach," in *IEEE International Conference on Network Protocol (ICNP)*, November 2003.

[15] Srdjan Capkun, Jean-Pierre Hubaux, and Levente Buttyan, "Mobility helps security in ad hoc networks," in *4th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'03)*, 2003, pp. 46–56.

[16] F. J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North Holland, Amsterdam, 1997.

[17] Yvo Desmedt and Yair Frankel, "Threshold cryptosystems," in *CRYPTO '89*, Giles Brassard, Ed. IACR, 1990, number 435 in LNCS, pp. 307–315.

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

[19] Torben P. Pedersen, "A threshold cryptosystem without a trusted party," in *EUROCRYPT '91*, D.W. Davies, Ed. IACR, 1991, number 547 in LNCS, pp. 552–526.

[20] Gabriel Montenegro and Claude Castelluccia, "Crypto-based identifiers (cbids): Concepts and applications," *ACM TISSEC*, vol. 7, no. 1, February 2004.

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

Contents

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399