Self-configurable Key Pre-distribution in Mobile Ad Hoc Networks

Claude Castelluccia[1⋆], Nitesh Saxena[2], and Jeong Hyun Yi[2⋆⋆]

[1] INRIA
655, avenue de l'Europe
38330 Montbonnot, France
claude.castelluccia@inrialpes.fr
[2] School of Information and Computer Science
University of California at Irvine
Irvine, CA 92697, USA
{nitesh,jhyi}@ics.uci.edu

Abstract. We present two new schemes that, in the absence of a centralized support, allow a pair of nodes of a mobile ad hoc network to compute a shared key without communicating. Such a service is important to secure routing protocols [1–3]. The schemes are built using the well-known technique of threshold secret sharing and are secure against a collusion of up to a certain number of nodes.
We evaluate and compare the performance of both the schemes in terms of the node admission and pairwise key establishment costs.

1 Introduction

Mobile ad hoc networks (MANETs) are, by their very nature, vulnerable to many types of attacks. The security of MANETs 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 MANETs, 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 [1–3]) cryptographic keys need to be established *prior* to communication. As a result, standard key exchange solutions, e.g., Station-to-Station protocol [4], 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 MANETs. Related to the latter is the underlying use of public key cryptography which is too expensive for some mobile devices.

Contributions: This paper proposes two efficient, fully distributed and secure key management solutions for MANETs. The solutions, collectively referred to as *Threshold Key Pre-distribution*, allow nodes in a MANET to establish pairwise keys without communicating and without the need of a PKI. The first scheme,

⋆ This work has been done while visiting UC Irvine.
⋆⋆ corresponding author

called Matrix Threshold Key Pre-distribution ($MTKP$), results from the blending of two well-known techniques: Blom's key pre-distribution [5, 6] and threshold secret sharing [7], and the second scheme, referred to as Polynomial Threshold Key Pre-distribution ($PTKP$), employs threshold secret sharing using a polynomial. In both $MTKP$ and $PTKP$, a node joins a MANET by receiving a secret token from t different nodes, where t is a security parameter. The schemes are *auto-configurable* in the sense that there is no centralized support required and a node becomes a member only if it is approved by at least t member nodes. Once a node becomes member, it can compute a secret key with any other member without interaction. The proposed schemes are secure against collusion of up to a certain number $(t-1)$ of compromised nodes.

The contribution of this paper is not limited to just the design of efficient key distribution schemes. We also demonstrate our claims of efficiency via extensive analysis and experiments. The schemes have been implemented and tested in a real MANET setting and their performance is compared and analyzed in detail.

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. Sections 4 and 5 present our Threshold Key Pre-distribution schemes $MTKP$ and $PTKP$ respectively. We discuss some security and other relevant issues of the proposed schemes in Section 6. Finally, in Section 7, we describe the implementation and the performance of our schemes.

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. Zhou and Haas [8] proposed to distribute a Certification Authority (CA) service among several nodes of the network. Although attractive, this idea is not applicable to MANETs. Their approach is hierarchical: only selected 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.

In a related result, Kong, et al. [9] developed an interesting Threshold-RSA (TS-RSA) scheme specifically geared for MANETs. Unfortunately, as pointed out in [10, 11], TS-RSA is neither verifiable nor secure. An alternative Threshold-DSA (TS-DSA) scheme [10] provides verifiability and, hence, tolerates malicious insiders. However, TS-DSA requires $2t-1$ signers to issue certificates, is heavily interactive and thus become quite inefficient in MANET settings. Moreover, all these solutions require a pair of nodes to perform key exchange protocol to establish shared keys.

Recently, Zhu, et al. [12] proposed a pair-wise key distribution scheme based on the combination of probabilistic key sharing and threshold secret sharing. 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.

3 Building Blocks

3.1 Threshold Secret Sharing

(t, n) threshold cryptography 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 up to $t-1$ parties can do so. We use Shamir's secret sharing scheme [7] 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 $f(x) = S + a_1 x + \cdots + a_{t-1}x^{t-1}$ 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 $ss_i = f(id_i) \pmod q$, 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 $l_i(0) = \prod_{j=1, j\neq i}^{t} \frac{id_j}{id_j - id_i} \pmod q$. To enable the verification of the secret shares, TD publishes a commitment to the polynomial as in *Verifiable Secret Sharing* (VSS) [13]. VSS setup involves a large prime p such that q divides $p-1$ and a generator g which is an element of $\mathbb{Z}_p^*$ of order q. TD computes w_i $(i = 0, \cdots, t-1)$, called the **witness**, such that $w_i = g^{a_i} \pmod p$ and publishes these w_i-s in some public domain (e.g., a directory server). On receiving the secret share ss_i from M_i, M_j verifies the correctness of ss_i by checking $g^{ss_i} = \prod_{k=0}^{t-1} (w_k)^{{id_i}^k} \pmod p$.

3.2 Blom's Key Pre-distribution

Blom proposed a key pre-distribution scheme that allows any pair of users in a group to compute a pairwise key without communicating [5]. 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 [14]. 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, the *key matrix* $K = AB$ is also symmetric

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

This shows that K is also a symmetric matrix.

Every user in the group is given a row (corresponding to its identifier) of the matrix A. The user M_i multiplies its row of the matrix A with the j^{th} column of the matrix B to establish a shared key with another user M_j.

4 MTKP: Matrix Threshold Key Pre-distribution

The different steps of the *MTKP* scheme, which is based on Blom's key pre-distribution described in previous section, are summarized as follows. (A more precise description of each of these steps is presented in the following subsections.)

1. *Bootstrapping:* The 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, $ss_i(D)$ for $i = 1, \cdots, n$, such that at least t shares are required to reconstruct it. Each node receives a share of D.
2. *Member Admission:* A prospective member M_η initiates the admission protocol by sending a `JOIN_REQ` message to the network. A member node, that receives this `JOIN_REQ` message and approves the admission of M_η, replies, over a secure channel (refer to Section 6), 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 receives shares from at least t different 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. Each joining node is also provided with *partial* shares of the whole matrix D from t current member nodes, which it can use to reconstruct its *share* of the matrix A, and consequently use it to admit new members.
3. *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 by a quorum of their peers. The 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.

4.1 Bootstrapping

In *MTKP* scheme, a network can be bootstrapped (i.e., initialized) by one node (centralized bootstrapping) or a set of t or more nodes (distributed bootstrapping).

Centralized Bootstrapping. The centralized bootstrapping proceeds as follows. First, a founding member FM generates the network parameters, namely N, λ, t, q, p, g, and the matrices $D = [d_{ij}]$ and $B = [b_{ij}]$, where N is the maximum numbers of nodes in the network, (λ, q, p, g) are the security parameters, B is $\lambda \times N$ public matrix such that $b_{ij} = (g^j)^i \pmod q$ for $i, j \in [1, \lambda]$, and D is $\lambda \times \lambda$ symmetric matrix of secrets.

Next, the FM publishes $(N, \lambda, t, q, p, g, B)$ in some public directory, but keeps D secret. It then computes the matrix A such that $A = (DB)^T$ and sends a share of the whole matrix A to each node.

To compute the share $ss_v(D)$ for member M_v, 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.

Distributed Bootstrapping. The network can alternatively be bootstrapped by a set of t or more founding members. The secret matrix D can be generated in fully distributed manner. Note that in the centralized mode, the FM is similar to a trusted third party and is, therefore, a single point of failure. 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. This procedure is so-called *Joint Secret Sharing (JSS)*. For more details, refer to [15].

4.2 Member Admission

In order to join the network, a prospective node M_η must collect at least t shares of matrix A's row η from the current member nodes and a valid share of the whole matrix D. Figure 1 shows the protocol message flow for the member admission process.[3].

1. M_η sends to at least t current member nodes M_ν-s ($\nu \in \{1, n\}$) a signed `JOIN_REQ` message which contains his identity id_η and his public Diffie-Hellman (DH) component $y_\eta (= g^{x_\eta} \mod p)$. The details about how id_η is generated and verified are discussed in Section 6.

[3] In order to secure the protocol against common *replay* attacks [4], 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.

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

Fig. 1. *MTKP* Admission Protocol

2. After verifying the signed `JOIN_REQ`, the member nodes who wish to participate in the admission process of M_η reply with a signed message containing their respective values id_ν and y_ν.
3. M_η selects t sponsors $M_\mu(\mu \in_R \nu, |\mu| = t)$, computes a secret key $DHK_{\eta\mu}$ with each of them, forms a sponsor list SL_η which contains the ID-s of the t selected sponsors, and replies with an authenticated acknowledgment message to each of them.
4. Each sponsoring node (M_μ) on 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 $DHK_{\eta\mu}$. Each (M_μ) also responds with the *shuffled* partial share of matrix D, $pss_\mu^\eta(D)$, such that $pss_\mu^\eta(D) = [pss_\mu^\eta(d_{ij})] = [ss_\mu(d_{ij}) \cdot l_\mu(\eta)] \pmod q$ for $i, j = 1, \cdots, \lambda$. This message is also encrypted using $DHK_{\eta\mu}$.

   Note that the Lagrange coefficients $l_\mu(\eta)$ are publicly known, and therefore, M_η can derive $ss_\mu(d_{ij})$ from $pss_\mu^\eta(d_{ij})$. This can be prevented using the *shuffling* technique proposed in [9] 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 nodes.
5. M_η decrypts the messages it receives from the different nodes and calculates his own $r_\eta(A)$ by adding up all $ss_\mu(r_\eta(A))$-s as follows: $r_\eta(A) = \sum_{\mu=1}^{t} ss_\mu(r_\eta(A)) \cdot l_\mu(0) = [\sum_{\mu=1}^{t} ss_\mu(r_\eta(a_{\eta j})) \cdot l_\mu(0)] \pmod q$ for $j = 1, \cdots, \lambda$. M_η also calculates his own share of the matrix D, $ss_\eta(D)$, by adding up the partial share values 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 $i, j = 1, \cdots, \lambda$.

4.3 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 $d_{ij} = d_{ji}$,

$$K_{ij} = \sum_{\beta=1}^{\lambda} a_{i\beta} b_{\beta j} = \sum_{\beta=1}^{\lambda} \sum_{\alpha=1}^{\lambda} d_{\beta\alpha} b_{\alpha i} b_{\beta j} = \sum_{\alpha=1}^{\lambda} \sum_{\beta=1}^{\lambda} d_{\alpha\beta} b_{\beta j} b_{\alpha i} = \sum_{\alpha=1}^{\lambda} a_{j\alpha} b_{\alpha i} = K_{ji}.$$

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

5 PTKP: Polynomial Threshold Key Pre-distribution

The various steps of the *PTKP* scheme, which is based on polynomial secret sharing, are summarized as follows.

1. *Bootstrapping:* The network is bootstrapped by either one single founding member or a set of founding members. The founding member(s) compute the secret share polynomial $f(z)$, defined in Section 3.1. Every member is provided with a share of this polynomial.
2. *Member Admission:* A prospective member M_η initiates the protocol by sending a `JOIN_REQ` message to the network. A member node, that receives this `JOIN_REQ` message and approves the admission of M_η, replies, over a secure channel (refer to Section 6), with *partial* shares of the polynomial which it can use to reconstruct its *share* of polynomial, and consequently use it to admit new members and establish pairwise keys with other members.
3. *Secret Key Computation:* Each node uses its secret share and the public VSS information (as described in Section 3.1) to compute pairwise keys with other nodes.

5.1 Bootstrapping

Centralized Bootstrapping. The centralized bootstrapping works exactly as described in Section 3.1.

Distributed Bootstrapping. A group of t or more founding members employ JSS [15] to collectively compute shares corresponding to Shamir secret sharing of a random value.

5.2 Member Admission

In order to join the network, a prospective node M_η must collect at least t partial shares from existing nodes to be able to compute its secret share. Figure 2 shows the protocol message flow for the member admission process.

$$
\begin{array}{|ll|r|}
\hline
msg1(M_\eta \rightarrow M_\nu)\text{: } REQ = \{id_\eta, y_\eta\},\ S_\eta(REQ) & (1)\\
msg2(M_\eta \leftarrow M_\nu)\text{: } REP = \{id_\nu, y_\nu\},\ S_\nu(REP, H(REQ)) & (2)\\
msg3(M_\eta \rightarrow M_\mu)\text{: } SL_\eta, MAC(DHK_{\eta\mu},\ H(SL_\eta, msg1, msg2)) & (3)\\
msg4(M_\eta \leftarrow M_\mu)\text{: } E_{DHK_{\eta\mu}}\{pss_\mu(\eta)\} & (4)\\
\hline
\end{array}
$$

Fig. 2. *PTKP* Admission Protocol

1-3. Steps 1-3 are exactly the same as in the *MTKP* admission protocol described in Section 4.2.

4. Each sponsoring node (M_μ) on receiving $msg3$, computes the secret key $DHK_{\eta\mu}$ and replies with the *shuffled* partial share[9], $pss_\mu(\eta)$, such that $pss_\mu(\eta) = ss_\mu \cdot l_\mu(\eta) \pmod q$. This message is encrypted using $DHK_{\eta\mu}$.
5. M_η decrypts the messages it receives from the different nodes and calculates his own secret share ss_η, by adding up the partial share values such that $ss_\eta = \sum_{\mu=1}^{t} pss_\mu(\eta)$.

5.3 Secret Key Computation

Any pair of nodes M_i and M_j can establish shared keys using their respective secret shares ss_i, ss_j and the public VSS information as described in Section 3.1. M_i computes $g^{ss_j} = \prod_{k=0}^{t-1}(w_k)^{id_j{}^k} \pmod p$ from the public commitment values, and exponentiate it to its own share ss_i to get a key $k_{ij} = (g^{ss_j})^{ss_i} \pmod p$. Similarly, M_j computes $g^{ss_i} = \prod_{k=0}^{t-1}(w_k)^{id_i{}^k} \pmod p$ and exponentiate it to its own share ss_j to get a key $k_{ji} = (g^{ss_i})^{ss_j} \pmod p$. Since, $k_{ij} = k_{ji}$, M_i and M_j have a shared secret key.

The above scheme remains secure under the *Computational Diffie-Hellman* (CDH)[4] assumption. In other words, an adversary who corrupts at most $t-1$ nodes, can not compute a shared key between any pair of uncorrupted nodes, as long as the CDH assumption holds.

6 Discussion

Identifier Configuration. In the *MTKP* and *PTKP* schemes, the identifier id_i of each node M_i must be *unique* and *verifiable*. Otherwise, a malicious node could use the identifier of some other node and get its secret from the member nodes during the admission process.

For unique and unforgeable ID assignment, we propose to use a solution based on *Crypto-Based ID (CBID)* [16]: 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 or DH public key y_i in our schemes, NID is the network identifier and $H_{64}(\cdot)$ a 64-bit long hash function. When a node contacts the member nodes for admission, it sends its identifier id_i together with its ephemeral public key PK_i and signs a challenge sent by the member node. Upon reception of the signature, the member node can verify that the id_i actually belongs to the requesting 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.

Secure Channel Establishment. In the proposed admission protocols, the channels between the node requesting admission and each of the member nodes must be authenticated and encrypted. It has to be authenticated because each member node 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 member node could send the shares to an impersonating node. Similarly, the joining node also

[4] CDH assumption: In a cyclic group generated by $g \in Z_p^*$ of order q, for $a, b \in \mathbb{Z}_q^*$, given $(g, g^a \pmod p, g^b \pmod p)$, it is hard to compute $g^{ab} \pmod p$.

needs to authenticate the member nodes. The channel 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 is 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 PKs that are sent in message 1 and 2 of the protocols described in Sections 4.2 and 5.2 do not need to be certified.

Parameters Selection. The security of the *MTKP* scheme relies on two security parameters t and λ, whereas the *PTKP* scheme depends only on t. λ and t denote the number of collusions needed to break these schemes. These parameters should be selected carefully. In particular, it is suggested to set $\lambda = t$. However, more generally, λ should be at least t in the hierarchical MANET settings where only a subset of nodes possesses the ability to admit new nodes. For the evaluation of our schemes (as described in the next section) we set $\lambda \geq t$.

DoS Resistance. A malicious node can easily launch a DoS (Denial-of-Service) 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 and its share of the whole matrix D in the *MTKP* scheme and its secret share in the *PTKP* scheme) before using them. This can be done with *Verifiable Secret Sharing (VSS)* [13], as detailed in an extended version of this paper [17].

7 Performance Evaluation

We implemented both *MTKP* and *PTKP* protocols and evaluated them in a real MANET environment in terms of node admission and pairwise key computation costs.

7.1 Experimental Setup

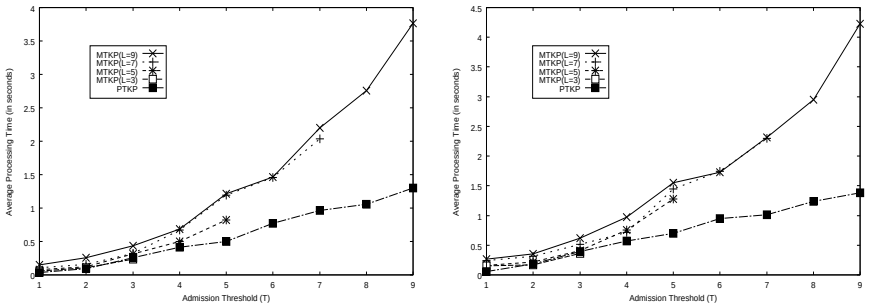
The *MTKP* and *PTKP* protocol suite is implemented on top of the OpenSSL library [18]. It is written in C for Linux, and consists of about 10,000 lines of code for each. The source code is available at [19].

For the experimental set-up, we used a total of five laptops; four laptops with a Pentium-3 800MHz CPU and 256MB memory and one laptop with a Mobile Pentium 1.8 GHz CPU and 512MB memory. Each device ran Linux 2.4 and was equipped with a 802.11b wireless card configured in ad-hoc mode. Specifically, for measuring the admission cost, four laptops with same computing power were used to configure the existing member nodes and the high-end laptop was used for the joining node. In our experiments, each node (except the joining node)

was emulated by a daemon and each machine was running up to three daemons. The measurements were performed with the different threshold values t and λ for *MTKP*. The size of the parameters q was set to 160-bits and p to 512-bits or 1024-bits.

7.2 Admission Cost

To evaluate the admission cost, we measured the total processing time between the sending of the `JOIN_REQ` by the prospective node and the receiving (plus verification) of acquired credentials (i.e., $r_\eta(A)$ and $ss_\eta(D)$ in *MTKP* and ss_η in *PTKP*). The resulting measurements include the average computation time of the basic operations, the communication costs such as packet encoding and decoding time, the network delay, and so on.



(a) $|p|$=512 (b) $|p|$=1024

Fig. 3. Admission Cost

Figure 3 shows the average admission time for the joining node for different values of the threshold t. For the *MTKP* testing, λ was set to 3, 5, 7 and 9. (In the figure, λ is denoted by L.)

As observed from the graphs, the cost for a node to join the network with *PTKP* is cheaper than that of *MTKP*. This difference in the costs between *MTKP* and *PTKP* is even higher for higher threshold values. The reason is quite intuitive: *MTKP* requires more computation and bandwidth than *PTKP*. More specifically, the *MTKP* scheme requires $O(\lambda^2 t)$ *multiplications* and $O(\lambda^2)$ *exponentiations* whereas *PTKP* requires only $O(t)$ *multiplications* and $O(1)$ *exponentiations*. For the bandwidth costs, refer to Table 1.

Table 1. Bandwidth Comparison

	MTKP	PTKP
Admission	$O(\lambda t\|q\|) + O(\lambda^2\|q\|)$	$O(t\|q\|)$
Shuffling	$O(\lambda^2 t^2\|q\|)$	$O(t^2\|q\|)$

This table shows that the *PTKP* scheme is very efficient in terms of bandwidth. This is an important property for MANET systems which consist of

battery-operated devices, because wireless transmission is considered as the most energy consuming operation.[5]

7.3 Key Computation Cost

Table 2 compares the cost of computing a pair-wise key in our schemes. The results show that *MTKP* performs significantly better than a *PTKP* protocol. The achieved gains with $\lambda = 9$ range from 10 ($t = 1$) to 13 ($t = 9$), and from 305 to 307 for 512-bit and 1024-bit p, respectively. In other words, *MTKP* is 10 to 307 times faster than *PTKP* when establishing a shared secret key.

Table 2. Key Computation Cost (in msecs, P4-3.0GHz, 1GB Memory)

	MTKP ($\lambda \geq t$)				PTKP	
t	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$	$\lambda = 9$	$\|p\| = 512$	$\|p\| = 1024$
1	0.0371	0.0301	0.0430	0.0550	0.574	17.780
2	0.0398	0.0415	0.0506	0.0570	0.683	18.150
3	0.0436	0.0424	0.0568	0.0564	0.713	18.180
4	-	0.0365	0.0595	0.0655	0.663	18.220
5	-	0.0431	0.0565	0.0629	0.753	18.370
6	-	-	0.0628	0.0563	0.772	18.450
7	-	-	0.0562	0.0629	0.782	18.570
8	-	-	-	0.0644	0.851	18.540
9	-	-	-	0.0637	0.871	19.120

These results were actually expected because in *MTKP* the pair-wise computation requires only $O(\lambda)$ modular *multiplications* where the modulus size is 160 bits. In contrast, *PTKP* requires $O(t)$ expensive modular *exponentiations* with a modulus size of 512 or 1024 bits.

8 Conclusion

We presented *Threshold Key Pre-distribution*: distributed solutions to the key pre-distribution problem in MANETs. Our solutions, *MTKP* and *PTKP*, are based on the secret sharing techniques and are secure against collusive attacks by a certain threshold of nodes. The solutions allow any pair of nodes in the network to establish shared keys *without communication*, as opposed to the standard Diffie-Hellman key exchange protocols. We implemented the *MTKP* and *PTKP* schemes and evaluated them in real MANET setting. Our analysis show that *MTKP* fares better than *PTKP* as far as the pairwise key establishment costs are concerned. However, in terms of the node admission costs, the latter outperforms the former. Based on this analysis, we conclude that the *MTKP* scheme is well-suited for MANET applications where node admission is not a frequent operation, whereas the *PTKP* scheme is more applicable for highly dynamic MANETs consisting of mobile devices with reasonably high computation power.

[5] It has been shown that sending one bit of data is roughly equivalent to adding 1000 32-bit numbers [19].

References

1. Hu, Y.C., Perrig, A., Johnson, D.B.: Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In: ACM MobiCom. (2002) 12–23
2. Hu, Y.C., Johnson, D.B., Perrig, A.: SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In: IEEE WMCSA. (2002) 3–13
3. Papadimitratos, P., Haas, Z.: Secure Routing for Mobile Ad Hoc Networks. In: SCS CNDS. (2002)
4. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press. (1997) ISBN 0-8493-8523-7.
5. Blom, R.: An Optimal Class of Symmetric Key Generation Systems. In: EUROCRYPT '84. LNCS, IACR (1984)
6. Leighton, T., Micali, S.: Secret-Key Agreement without Public-Key Cryptography. In: CRYPTO'93. (1993)
7. Shamir, A.: How to Share a Secret. Communications of the ACM **22** (1979) 612–613
8. Zhou, L., Haas, Z.J.: Securing Ad Hoc Networks. IEEE Network Magazine **13** (1999) 24–30
9. Kong, J., Zerfos, P., Luo, H., Lu, S., Zhang, L.: Providing Robust and Ubiquitous Security Support for MANET. In: IEEE ICNP. (2001)
10. Narasimha, M., Tsudik, G., Yi, J.H.: On the Utility of Distributed Cryptography in P2P and MANETs: The Case of Membership Control. In: IEEE ICNP. (2003) 336–345
11. Jarecki, S., Saxena, N., Yi, J.H.: An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol. In: ACM SASN. (2004) 1–9
12. Zhu, S., Xu, S., Setia, S., Jajodia, S.: Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach. In: IEEE ICNP. (2003)
13. P.Feldman: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: FOCS. (1987) 427–437
14. MacWilliams, F.J., Sloane, N.: The Theory of Error-Correcting Codes. North Holland, Amsterdam (1997)
15. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In Davies, D., ed.: EUROCRYPT '91. Number 547 in LNCS, IACR (1991) 552–526
16. Montenegro, G., Castelluccia, C.: Crypto-based Identifiers (CBIDs): Concepts and Applications. ACM TISSEC **7** (2004)
17. Castelluccia, C., Yi, J.H.: DoS-Resistant Self-Keying Mobile Ad-Hoc Networks. Technical report, INRIA-5373, http://sconce.ics.uci.edu/gac/publication.html (2004)
18. OpenSSL Project: (`http://www.openssl.org/`)
19. Peer Group Admission Control Project: (`http://sconce.ics.uci.edu/gac`)