Towards Integrating Key Distribution with Entity Authentication for Efficient, Scalable and Secure Group Communication in MANETs

Maria Striki, John S. Baras
Electrical and Computer Engineering and
the Institute for Systems Research
University of Maryland, College Park
College Park, Maryland, 20740, U.S
mstriki@isr.umd.edu, baras@isr.umd.edu,

Abstract— **In this paper we focus on the design of key management (KM) schemes tailored for the environment of Mobile Ad Hoc Networks (MANETs). A MANET is a collection of wireless mobile nodes, communicating among themselves over possibly multi-hop paths, without the help of any infrastructure such as base stations or access points. The fact that no central authorization entity is assumed at all times for all nodes makes the task of network operations more difficult and indicates the need for distributed algorithms to provide the functions of centralized entities. KM ensures communication security among nodes and the capability of their cooperation as a secure group. It consists of key generation, user authentication and key distribution services. In this work we address key distribution, group key generation, entity authentication: we emphasize that entity authentication should be designed with key distribution algorithms in mind and vice versa, to achieve efficient and scalable KM schemes for MANETs. We present an entity authentication scheme based on the Merkle Tree algorithm, applied on a key generation protocol recently developed – MOT- to produce an efficient, scalable and secure KM scheme.**

Keywords; MANET, key generation, authentication, Merkle Trees

I. INTRODUCTION

As the development of wireless multicast services such as cable TV, secure audio and conferencing, visual broadcasts, military control, grows, research on security for group communication becomes increasingly important. There is demand consequently for more efficient KM schemes, suitable for wireless mobile networks. Moreover, collisions, low link quality and other factors result in unreliable links or excessive delay in MANETs. All these network constraints should be anticipated for the design of a scalable, fault-tolerant KM scheme that operates securely and efficiently in MANETs. In our work, we focus on the design of group key distribution and authentication techniques in networks of limited bandwidth, unreliable channels, where topology is changing fast. Network nodes may have limited capacity, computation, transmission power (satellites, laptops, PDAs, cell-phones). We show how **entity authentication** can be merged with key distribution techniques to produce a **powerful**, **efficient**, **autonomous** KM scheme (in particular we present a **Merkle Tree based** entity authentication scheme combined with an existing key distribution protocol to reduce the additional KM overhead).

Most of the current KM schemes are designed for wire-line networks that are free from most of the constraints of MANETs. Group KM functions, robust to tolerate frequent node failures, network partitions/merges, delays in critical messages, extensive computations etc., are required. Also, a change in the topology of a group might occur while the group key is being calculated. In some protocols re-keying may cause enormous overhead, as the key establishment operation must start over, and nodes might have to be re-authenticated. These constraints make it hard for most group KM schemes to catch up with the rapidly changing topology of the network.

For secure communications, there is no point in distributing shared or session keys if we cannot verify the identity of the participant nodes. Thus, entity authentication should occur prior to key distribution. Key distribution is usually more sensitive to the deployment of nodes within the network, and thus considered a much heavier operation in terms of bandwidth and processing than entity authentication. So, it would be more appropriate to decide first on suitable key distribution protocols, and then on entity authentication schemes. Extensive research has been done in terms of node to node authentication. Such algorithms are rather prescribed and do not provide as much freedom as key distribution schemes. One may argue that incorporating existing authentication schemes, to selected key distribution protocols should be very simple. Next, we will briefly justify why entity authentication demands a substantial merit of our investigation.

The **environment of interest** in this work is this of **Flat MANETs:** *few, if any trusted special entities exist, that may not be accessible to all nodes at all times*. We want key distribution schemes to be *fault-tolerant* in the following aspects: the group leader should not be single point of failure, the protocol should recover from member failures during key establishment and tolerate group partitions/merges at any time during a session. We want to investigate the desired properties of such protocols to operate well in this framework. To this end we classified existing protocols in two families: *contributory* where all members take equally part in key generation, and *non-contributory*, where usually a group leader distributes the group key alone. In previous work we studied the limitations of these two families in MANETs, for the design of a fault-tolerant key distribution protocol.

In the case of **non-contributory** protocols, a multi-tasking group leader is a single point of failure. Leader failure may occur quite often in MANETs, and this is the weak point of non-contributory protocols in comparison to contributory ones. Finding group members to replace a faulty leader is not enough. The new leader should securely, quickly, and with low overhead, obtain all information gathered by the previous leader up to that point which is not always easy. It would be preferred that the role of the leader is rather coordinating, as in contributory schemes, storing information that can be easily retrieved by a next leader (*e.g. TGDH*). To reduce group partitions and frequent leader elections, parameters such as mobility, processing capabilities of a node etc. should be considered. In tree-based **non-contributory** protocols, in the event of a node failure, a new group key is computed, updating only a restricted number of keys. On the other hand, the strongest point of non-contributory schemes is that members' contributions for establishing a key are independent and need not follow a strict ordering. In the event of failures or delays, the rest of the nodes proceed normally to key establishment. In a **contributory** protocol (e.g. GDH.2), each member is expected to contribute its sub-share according to a defined pattern. If a node does not respond on time, the whole procedure comes to a standstill as all further actions depend now on this contribution and the process starts all over again. No node constitutes a single point of failure in these schemes. Considering these, we claim that the hybrid **2^d-Octopus** (O)- based on **Hypercube** - is well-suited for MANETs: it tolerates various kinds of failures or resumes with low overhead. Observing this, we designed two new hybrid protocols based on (O): (GDH.2-based **MO** and TGDH-based **MOT**). MO and particularly MOT are more efficient than the original (O). MOT has very satisfactory overall performance and is very robust as seen in previous work [5]. We have thus chosen to design authentication algorithms with (O) and MOT in mind.

Our main concern in selecting entity authentication algorithms is that the resulting KM scheme operates efficiently as a whole. We should not design efficient independent solutions for each service, but rather exploit any kind of redundancies that might result from their integration, further reducing the total cost. Regarding the above, certain issues arise. Is it preferable to design re-authentication based on credentials the member initially or progressively obtains? Which members should (re)-authenticate a node: the group leader, an arbitrary subset of members? According to one scenario, the member that first contacted the node should be authorized to execute these operations, based on its partial view of trust. According to another scenario, the policy might require that all group members vote (global view of trust) to accept or not the node in the group. The issue of low overhead *vs.* trust and security must be investigated.

The resources available during network bootstrapping significantly determine the appropriate entity authentication scheme for the first time at least, with the potential to switch to lighter methods after the appropriate security associations are established. The unavailability of CAs, makes it difficult to obtain correct public keys (PKs) for the parties involved - on which most authentication algorithms rely. So, the first issue concerning authentication is to develop dynamic mechanisms to substitute the functionality of centralized CAs. We still then have to decide on the most appropriate node to node authentication method, based on the metrics that we are mostly interested to reduce. A seemingly suitable solution would be to use *id*-based (IBE) schemes. It turns out that it can be reduced to distributing the functionality of an absent centralized entity, so it is no better than the already stated approaches.

A subtle problem that KM schemes do not handle is the following: a compromised member, once it gets the session key, can use it to mislead other nodes in terms of the message originator. KM does not provide data authentication and unless it uses proper mechanisms to detect compromised nodes, this attack could disrupt its correct function. A proactive method based on Merkle Trees [15] has been proposed to alleviate this problem. We propose a method to construct a distributed dynamic CA well-suited for MANETs, based on KM groups and on Merkle Trees (MTs). It is easy to understand now why the topic of entity authentication for KM in MANETs should be thoroughly investigated.

II. PREVIOUS WORK

Becker et al.[1], derived lower bounds for contributory key distribution and proved them for DH protocols. Steiner et al.[2], extended DH protocols to groups. GDH.2 acquires implicit entity authentication. TGDH by Kim et al. [3], is a new hybrid protocol that blends binary key trees with DH. Becker in [1], introduced Hypercube as one requiring the min. number of rounds. In [4], Asokan et al. added recovery from node failures. Becker introduced Octopus that required min. number of messages and then 2^d-Octopus that combined Octopus with Hypercube to an efficient protocol for arbitrary number of nodes. Non-contributory protocols are based on a key distribution center. The simplest is GKMP. Logical Tree Hierarchy (LKH), creates a hierarchy of keys for members. Efficient evolution is One-Way Function Tree (OFT) [6] that minimizes the total bits broadcast after a membership change.

A large number of private or public entity authentication schemes currently exist (*RSA, Schnorr* etc.*)*. Lamport invented the *one-time password scheme* [7] that uses hash chains to prevent certain attacks. Bellovin et al. created the *first strong password* protocols. *Fiat-Shamir* efficiently transforms a public key identification to a signature scheme. In Kerberos, two entities authenticate each other via a globally trusted CA. PGP follows a web-of-trust model that does not scale beyond a relatively small community of trusted nodes. SUCV [8], binds a node's public key to its IP so that given its public key, correctness of IP is instantly verified (many drawbacks). An approach to deal with the absence of a CA is to distribute its functionality among a fixed group of servers, according to notions of threshold cryptography and proactive secret sharing [9], where a small group with rich network connectivity is assumed. More efficient proposals are those of Zhou [10] and Luo [11] that attempt to extend [9] to MANETs. For data authentication, Lamport, Merkle etc. created and refined digital signatures using one-way functions. Perrig et al. contributed to the design of authenticated broadcasts, introducing protocols like: BIBA, TESLA [12], SPINS etc. based on hash chains, loose time synchronization. Tree-authenticated values or *Merkle trees* [13], provide low storage overhead, and are used in [23], for hop-by-hop authentication. Bobba et al.[14] ensure entity authentication by using SUCVs.

III. PROTOCOLS USED TO CONSTRUCT MOT

We briefly describe the protocols used to construct MOT. They are documented in detail in the TR [5].

1. *Octopus Protocol*

Four parties A, B, C, D generate a group key by four key exchanges (KE). First, A-B, then C-D do a DH KE generating keys α^{ab}, α^{cd}. Then, A-C as well as B-D do a DH KE using as secret values the keys generated in the 1st step. A(B) sends $\alpha^{\phi(a^{ab})}$ to C(D), which sends $\alpha^{\phi(a^{cd})}$ to A(B) so that A-C (B-D) generate the joint key $\alpha^{\phi(a^{cd})\phi(a^{ab})}$. $P_1, P_2, \ldots, P_{n-4}$, A,…, D generate a common group as follows: A,…, D take charge of central control. The rest distribute themselves into 4 groups: $\{P_i \mid i \in I_A\}, \ldots, \{P_i \mid i \in I_D\}$, $I_A, \ldots, I_D$, are pair-wise disjoint. $P_1, \ldots, P_n$ generate a group key as follows:

1. $\forall X \in \{A, \ldots, D\}, \forall i \in I_X$, *X* generates joint key k_i with P_i via DH KE.

2. A, B, C, D do 4-party KE using values: $a=K(I_A), \ldots, d=K(I_D)$, where $K(J) := \prod_{i \in J} \phi(k_i)$, $J \subseteq \{1, .., n-4\}$ and hold the joint key $K = a^{\phi(a^{K(I_A \cup I_B)})\phi(a^{K(I_C \cup I_D)})}$.

3. The step is described for *A*. Parties *B, C, D* act accordingly. $\forall j \in I_A$, *A* sends 2 values to P_j: $a^{K(I_B \cup I_A \setminus \{j\})}, a^{\phi(a^{K(I_C \cup I_D)})}$. P_j derives $(a^{K(I_B \cup I_A \setminus \{j\})})^{\phi(k_j)} = a^{K(I_A \cup I_B)}$ & $K = a^{\phi(a^{K(I_C \cup I_D)})\phi(a^{K(I_A \cup I_B)})}$

2. Hypercube Protocol

2^d parties agree on a key within *d* rounds via DH key KE on the edges of a *d*-dimensional cube. In round 1, every participant *v* generates random number r_v and executes DH KE with participant $v+b_1$ using values r_v and r_{v+b1}. In round *i*, every participant *v* does a DH KE with $v+b_i$. Both parties use as secret the value generated in round *i*-1.

3. Tree Group DH (TGDH) protocol

TGDH acquires the binary tree structure. The leaves are the members and the group key is associated with the root. Any member can be leader at any point. Node *x* is associated with secret key k_x and blinded key $k_x' = g(k_x)$; *g* is a one-way function. Each member knows the un-blinded keys on its root path, and all blinded keys of the tree. Members compute un-blinded keys along their path. If a blinded key changes and the appropriate member gets the new value, it re-computes the keys on its path to find the new key. The secret key of an internal node is the result of a DH KE between its offspring. Initially all members becomes sponsors [5].

4. 2^d*- Octopus Based Protocol (O)*

For a group of arbitrary size, Octopus is generalized. In (O), 2^d parties now take charge of the central control. The remaining n-2^d parties divide into 2^d groups. (MO), (MOT) are based on GDH.2, TGDH respectively: they maintain the 2nd step of (O) intact and substitute the centralized scheme of the 1st and 3d step with GDH.2 or TGDH. The subgroup key becomes: $a^{ab\cdots z} = a^N$ (MO) or $a^{xy} = a^N$ (MOT). During the 1st step each subgroup establishes its own key and handles membership changes as indicated by GDH.2 and TGDH. In both protocols the subgroup key acquires the desired structure (not all protocols are appropriate to incorporate in (O)).

IV. BRIEF DISCUSSION ON RESULTS FOR MOT

A reactive algorithm [4] and a proactive one [5], have been added to Hypercube, to allow recovery from node failures, introducing either significant bandwidth overhead or extra computation overhead. Papers on (O) do not address cases of membership change. We analyzed all cases and calculated the cost values for (O), MO, and MOT. Hypercube is the core for all Octopus-based schemes, but the key generation scheme of each subgroup is important as well: Subgroups in (O) and (MO) assume either a single fixed leader or strict ordering during key establishment (GDH.2), cannot tolerate delays or failures and are not fault tolerant. TGDH however, assumes that any node should be ready to become sponsor and that the same amount of information is stored in all members with no considerable overhead. Since the subgroup is relatively small in size now and is deployed on a restricted area of the network too, TGDH can be considered more robust for flat MANETs.

Each "edge" of Hypercube represents a subgroup leader. The subgroup keys construct the initial secret share for Hypercube, which is used to calculate the group key. The subgroup leaders distribute parts of the group key to their members so that only they are able to reconstruct it. The subgroup key generation protocols may be selected with freedom. Each subgroup is deployed on a relatively *restricted* network area and it is easier to handle locally. Arbitrarily many members can be assigned to each subgroup, given the topology of the network. This results in less routing traffic and less bandwidth consumption. Also, a faulty leader can be replaced by another node from its own subgroup. It is clear how these properties render the protocol robust to node failures, group merges/partitions.

Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not withstanding any copyright notation thereon

V. Authentication Issues & Key Generation

A. Simple and Inexpensive Entity Authentication in MANETs

Lamport's Hash is a password protocol for wire-line networks. It achieves simple, low cost authentication, prevents replays, eavesdropping, impersonation attacks, with successive hashes. *Alice* and Bob share a password: *pwd*. Initially, Alice selects number *n*, computes $x_n=H^n(pwd)$, and sends Bob values $<n,x_n>$. When *Alice* wishes to prove her *id* to *Bob*, *Bob* sends $<n>$. *She* computes and sends *Bob* value $x_{n-1} = H^{n-1}(pwd)$. *Bob* compares $H(x_{n-1})$ to the value currently stored. Upon verification, *Bob* replaces $<n,x_n>$ with $<n-1,x_{n-1}>$ for a future re-authentication.

To apply this scheme to MANETs, and achieve mutual authentication, we modify it as follows: at bootstrapping, nodes execute an authenticated DH KE (e.g. sign the DH values they send), to obtain a secret key. They use this key once, in order to exchange initial quantities $<n,x_n>$, $<m,x_m>$ with a challenge response protocol for mutual authentication that requires low number of exchanges. From now on, each time they request re-authentication, they apply the original *Lamport*. This method has several advantages particularly if combined with protocols (O) or MOT: a subgroup leader may contribute the same value computed only once for its share in the DH KE. The members will send their individual shares $<n,x_n>$ to the leader, along with key distribution information, so that no extra bandwidth is required for this action. After bootstrapping, the leader can easily re-authenticate any subgroup member. If a member moves to another subgroup it need not re-initiate Lamport with the new subgroup leader: the previous leader communicates the "authentication state" of the member to the rest of subgroup leaders. This method requires public keys (PKs) for the initial phase. Our approach on how nodes get and trust other nodes' PKs, is discussed next.

B. Bootstrapping - Certification Authorities (CAs)

Preloading nodes with the appropriate information prior to deployment, becomes especially important in the absence of on-line CAs. To go around this problem, a node could use *SUCV*s to bind its IP with its PK at bootstrapping only at least. An obvious problem of this approach is that when a node moves to another LAN, it must comply with the auto-configuration hierarchy and may not be able to retain its IP and consequently its PK. Also, when the PK of the node must be refreshed, its IP should change, which is impractical. If the node relies only on off-line CAs, that bind its PK with its *id*, then time-sensitive certificate renewal/revocations, cannot be handled in a timely manner. Furthermore, how do nodes that have initially "spoken" to different off-line CAs "recognize" each other and form initial security associations? How to replace the functionality of CAs after deployment?

We address the first question by assuming that all nodes initially carry a certificate issued by an off-line CA which also includes a *key ring* that contains a predefined number of PKs of other trusted off-line CAs at random. Assume that node *A* wants to securely communicate with node *B*. If *A* knows the correct PK of the CA that authenticated *B* vice versa, then the two nodes will get the correct PK for each other and execute a public identification scheme. Otherwise, each node attempts to discover and authenticate nodes that have spoken to one of the CAs designated by its key-ring, with the goal to find a "certification path" that leads to the PK of the CA it is looking for. *A* utilizes the key-ring of *B* upon verification only. During the discovery process, a node obtains the PKs of several CAs. This approach is similar to [16], which we can be used to determine optimal parameters, e.g. length of the key-ring etc., to ensure that the certification paths are relatively small.

C. Simple and Inexpensive Data Authentication in MANETs

As we mentioned, Merkle Trees were used in [15] to provide data authentication. In brief, the scheme works as follows: to authenticate a sequence of *n* values $u_{0,},\ldots, u_{w-1}$ the sender virtually places them at the leaves of a binary tree, and blinds them with hash function *H*, so that $u_i^{'} = H(u_i)$. The value of any internal tree node *p* comes from its offspring *l, r* so that $m_p = H(m_l|m_r)$. The root authenticates all leafs. To ensure authenticity and non-repudiation, it is signed by the sender. To authenticate value u_i the sender discloses i, u_i, the signed root and all required values (sibling to those in the path to the root) that allow the receiver to reconstruct the root value and verify that it equals the expected value. It is not possible for another group member to impersonate the sender. This method is more efficient than signatures or TESLA since it is based on hashing, and requires only one signature per tree. The sender can pre-compute the appropriate $2n$ hashed values. The only price to pay is this extra bandwidth. Next, we will show how to use this concept to address our second question: create dynamic, distributed "CAs" from nodes that belong to a KM group.

D. Modified Merkle Trees for dynamic CA Construction

Assume a KM scheme that uses MOT for key generation. Subgroups form virtual binary trees. A CA per subgroup will be created, and initially all members will participate to the CA construction: each subgroup member creates its secret share m_i via DH KE with the sponsor. The sponsor uses Merkle Trees to authenticate these values with the unique root value. At this point we suggest a slight variation to the previous scheme: assume that the offspring of the root produce values *L* and *R*, known both only to the sponsor, which broadcasts values: g^L and g^R to the subgroup. It also sends the proper hashed values to members, so that each may reconstructs only *L* or only *R*. Through a DH KE, all members are able to calculate the value m_{root}: g^{LR}. This value will be used as the CA PK, whereas the CA SK $<LR>$ will be known only to the subgroup leader. The CA PK will be communicated to the rest of nodes.

The CA of a subgroup is robust to membership changes and certificates may still be valid, as long as there still exists an adequate amount of members having initially participated to the creation of the CA, and the subgroup leader that owns the CA SK is still the same. In this variation, the evicted node does not know the CA SK, and thus cannot tamper with its operations. If a node moves to another subgroup, it will eventually need to get a certificate from the new subgroup (smooth handoff) so that operations like update/revocation are faster, "localized", and cause less bandwidth overhead. The scheme is *very robust to failures*, but has certain *drawbacks*: it takes only two colluding members to reconstruct the CA SK, the sponsor knows the CA SK and is thus a single point of failure. We may also distribute the knowledge of the CA SK to a larger members' subset. Further details can be found in our TR [17]. Our method combines attributes of KM groups with the low cost Merkle Tree scheme to construct efficient and robust CAs. We wish to avoid existing impractical heavy bandwidth-delay solutions for MANETs (e.g. based on Threshold Crypto etc.). Future research focuses on the design of more efficient "authentication" schemes, on determining appropriate entities to participate, and on the design of distributed ways to monitor and collect "state information".

VI. RESULTS AND DISCUSSION

Cost	(O)+Mod. Lamport	MOT+Mod. (Lamport+Merkle)
GSC Store	$K(\lceil (n-2^d)/2^d \rceil + d)$ or $K(3\lceil (n-2^d)/2^d \rceil + d+1)$	$(\lceil n/2^d \rceil + 2\log\lceil n/2^d \rceil + d + \lceil (n-2^d)/2^d \rceil + 2)K$
Memb. Store	$(2+d)K+(X+1)K$	$(\lceil n/2^d \rceil + 2\log\lceil n/2^d \rceil + d + 2 + X)K$
Initial GSC Comp.	$(2\lceil (n-2^d)/2^d \rceil + 2d)C_E + (\lceil (n-2^d)/2^d \rceil)^{4/3} + 1.25(\lceil (n-2^d)/2^d \rceil)K^2 + \lceil n/2^d \rceil C_{rr} + (C_E + C_{PD}) + \lceil (n-2^d)/2^d \rceil (C_E+C_{PE}) + C_{SD} + 30K$	$(2\log\lceil n/2^d \rceil + 2d)C_E + \lceil n/2^d \rceil C_{rr}$ max. $+(C_E+C_{PD}) + \lceil (n-2^d)/2^d \rceil (C_E + C_{PE}) + C_{SD} + 30K$ (Lamport) $+60\lceil n/2^d \rceil K + 3C_E(2K) + (\lceil n/2^d \rceil - 3) C_{SE}(2K)$ (Merkle)
Initial Comm.	$(2n+(d-1)2^{d+1} + 2\lceil n/2^d \rceil)K$	$(2^d 2\lceil n/2^d \rceil + 2^{d+1}d + 6\lceil n/2^d \rceil)K$

Table1: Key Generation Costs after addition of authentication algorithms

Table1 shows some of the comparison results of (O) and MOT key generation schemes: an entity authentication scheme (Modified Lamport) has been added to both, and our modified Merkle Scheme for authentication and distributed CA services has been added only to MOT. It is interesting to observe that MOT continues to have the lowest initial computation cost despite the two authentication services it provides. As of communication cost, (O) performs slightly better for smaller d, but as d increases MOT becomes more and more efficient.

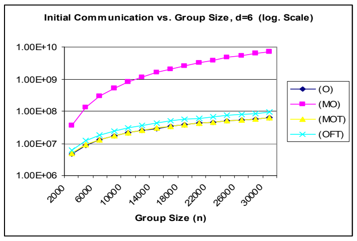

Figure1. Initial Comm/tion vs. Group Size, d=6. Non authenticated MOT, (O) achieve almost same lowest cost, OFT is slightly worse.

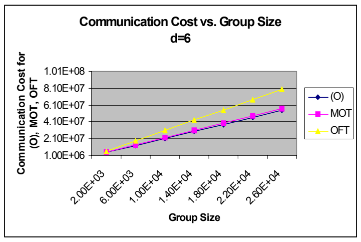

Figure 2. Initial Comm/tion vs. Group size, d=6, for authenticated (O), MOT compared to OFT. O & MOT achieve almost the same lower cost

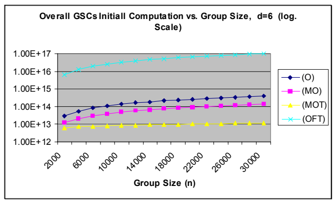

Figure 3. Previous results on Init. GSC comp. cost w/o authentication.

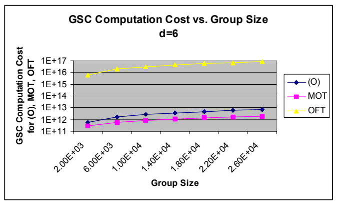

Figure 4. GSC computation cost of authenticated (O), MOT.

VI. SUMMARY AND CONCLUSIONS

We show how "authentication" can be merged with key distribution to produce efficient KM schemes in MANETs. We present a modified entity authentication scheme and construct dynamic distributed CAs, based on Merkle Trees. We integrated them with key generation protocols (O), MOT, and evaluated the results. We observed that MOT could still outperform the protocols it had been compared to before. The results underline the importance of combining KM techniques to exploit redundancies, and show that there is scope for further improvement. Research on the design of efficient, secure and robust "authentication", is still on-going.

REFERENCES

[1] Klaus Becker, Uta Wille. Communication Complexity of Group Key Distribution. Proc.5th ACM Conference on Computer & Comm/tions Security, pages 1-6, San Francisco, CA, November 1998. ACM Press

[2] Steiner M., Tsudik G., Waidner M., Diffie-Hellman. Key Distribution Extended to Groups. 3rd ACM Conference on Computer & Comm/tion Security, ACM Press, 1996.31-37

[3] A.Perrig. Efficient Collaborative Key Management Protocols for Secure Autonomous Group Communication. Int'l Workshop on Cryptographic Techniques E-Commerce CryptTEC'99.

[4] Asokan, Ginzboorg. Key-Agreement in Ad-Hoc Networks. Elsevier'00

[5] M. Striki, J. Baras. Efficient Scalable Key Agreement Protocols for Secure Multicast Comm/tion in MANETs. CSHCN TR 2002.

[6] D. McGrew. A.T.Sherman. Key-Establishment in Large Dynamic Groups Using One-Way Function Trees. May 1998.

[7] L. Lamport, "Password Authentication with Insecure Communication," Comm/tions of the ACM, Vol 24, No 11, November 1981, pp 770-772

[8] Montenegro, Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," NDSS 2002

[9] A. Shamir, "How to Share a Secret," Communications of ACM, 1979

[10] Zhou, Haas. "Securing adhoc networks", IEEE Networks, 13:24-30, '99

[11] J.Kong, P.Zerfos, H.Luo, S.Lu and L.Zhang. "Providing robust and Ubiquitous security support for MANET," IEEE ICNP 2001, 2001

[12] A.Perrig, R.Canetti, D.Song, J.D.Tygar.,"Efficient and Secure SourAuthentication for multicast,". Procs of the Symposium on Network and Distributed Systems Security (NDSS 2001), pages 35-46. Feb. 2001

[13] R.C.Merkle.,"Secrecy, Authentication and Public Key Systems,". Technical Report, Stanford University, June 1979

[14] R.Bobba, L.Eschenauer, V.Gligor, W.Arbaugh.,"Bootstrapping Security Associations for Routing MANETs,". TR, UMD, 2002-44, 2002

[15] A.Perrig, Y.C.Hu, D.B.Johnson,"Packet Leashes: A defense against Wormhole Attacks in Wireless Ad Hoc Networks", Infocom 2003

[16] L.Eschenauer, V.Gligor., "A Key Management Scheme for distributed sensor networks"., Procs of ACM CCS'02, pages 41-47, Nov, 2002

[17] M. Striki. J. Baras," Towards integrating key distribution with entity authentication for efficient, scalable and secure group communication in MANETs",CSHCN,TechnicalReport,2004