Leveraging a Minimal Trusted Computing Base for Securing On-Demand MANET Routing Protocols

Vinay Thotakura, Mahalingam Ramkumar
Department of Computer Science and Engineering
Mississippi State University, MS.

Abstract—**We propose an approach to secure on-demand mobile ad hoc network (MANET) routing protocols by leveraging low-complexity trustworthy MANET modules (TMM). Such TMMs, which perform some trivial hard-wired functions involving simple logical and hash operations, can be realized at low cost. We describe the functionality of such TMMs to offer a sound trusted computing base (TCB) for securing MANETs. We outline an approach to secure two popular on demand protocols - the ad hoc on demand distance vector (AODV) protocol, and the dynamic source routing (DSR) protocol - by leveraging the TCB.**

I. Introduction

A mobile ad hoc network (MANET) is constituted by battery operated mobile computers with limited wireless transmission range. A MANET routing protocol is a set of rules which dictate the actions to be performed by every node to enable any two nodes in a subnet to establish multi-hop paths and relay data packets.

In practice, some nodes may be under the control of a malicious users who may deliberately violate rules with the intention of disrupting the operation of the MANET or for selfish purposes; some nodes may misbehave accidentally due to hardware failure, or bugs in the MANET software, or even in the operating system of the mobile computer.

A. Trusted Computing Base

Securing a MANET involves providing some tangible assurances that nodes will abide by rules. Realizing assurances towards securing any system is achieved by amplifying the trust in a trusted computing base (TCB) [3]. Most strategies to secure MANETs employ cryptographic authentication of routing data to limit an attackers ability to disseminate inconsistent routing information. The TCB for facilitating cryptographic authentication includes a set of cryptographic algorithms (which are assumed to be unbreakable), and a trusted authority (TA) who distributes cryptographic material to all nodes of a MANET network. Secure MANET protocols that leverage this limited TCB i) fail to provide some important assurances, and ii) typically impose substantial overhead for resource limited battery operated mobile devices.

1) Contributions: In this paper we propose a simple and efficient TCB for MANET nodes which can be leveraged to improve the performance of MANETs by i) providing assurances that reduce the scope of attacks that can be launched by attackers, and by ii) reducing the overhead required for leveraging the TCB. In the proposed approach simple TCB functions are executed inside trustworthy MANET modules (TMM) housed in every MANET node. We assume that only the TMMs are trusted: the rest of the node - all other hardware and software - are untrusted.

An important prerequisite for a trustworthy module to *warrant* trust is that the TCB functions executed inside the module are simple, and consequently, easily verifiable. Simple TCB functions can also be implemented as hardwired logic (software-free), thus rendering moot a wide range of attacks that attempt to modify software. It is also desirable that the modules consume as little power as possible, and consequently disseminate negligible heat, as such modules can be physically well shielded from deliberate and accidental intrusions.

With these self-imposed limitations on TCB functions aimed at improving the reliability of TMMs while simultaneously lowering their cost, we seek a set of simple TCB functions for MANETs. While the proposed approach is general enough to be applicable to many different MANET protocols, in this paper we restrict ourselves to two popular on-demand protocols - ad hoc on demand distance vector (AODV) [1], and dynamic source routing (DSR) [2] protocol.

B. Overview of TMM Functionality

In the proposed approach TMMs housed in MANET nodes. TMMs are capable of establishing pairwise secrets with each other using strategies which demand low overhead for operations to be performed inside the trusted boundary of TMMs. These pairwise secrets are used for computing message authentication codes (MAC). Nodes communicate with their TMMs using fixed and well-defined interfaces - by writing into the input registers of the TMM and reading time-stamped MACs from the output registers of the TMM. Such MACs accompany MANET routing packets sent by nodes.

The trust in the TMM - viz., that i) the secrets protected by TMMs cannot be exposed, and ii) the simple functionality of the TMMs cannot be modified is amplified to secure AODV and DSR.

C. Organization

The rest of this paper is organized as follows. In Section II we begin an overview of AODV and DSR and explain the notions of trusted computing base (TCB) and trustworthy computing modules. In Section III we outline the TCB functions of TMM i) UpdateNT(); ii) RelayRR (); and iii) RelayData (), all of which output time-stamped MACs computed using secrets protected inside the TMM. In Section IV we explain how the three TCB functions can be utilized to provide some useful assurances regarding MANETs employing AODV or DSR as the routing protocol. The proposed approach to realize secure AODV and DSR is compared with other secure AODV/DSR approaches in the literature. Conclusions are offered in Section V.

II. Background

MANET routing protocols can be broadly classified into proactive and reactive protocols. In the former, nodes strive to maintain a consistent view of the subnet topology. In the latter topology information is acquired on demand. Popular examples of the former category include the dynamic source routing protocol (DSR) and ad hoc on demand distance vector (AODV) protocol. An example of the latter category is the dynamic sequenced distance vector (DSDV) protocol.

A. AODV and DSR

AODV is an ad hoc on-demand routing protocol where the distance to a destination is stored in a node's routing table as a routing record (RR), indexed using the destination identity. Other fields in the RR include a sequence number, hop-count, next-hop to reach the destination, and validity time.

When a node desires to communicate with a destination, and finds that it does not have a fresh route to the destination, a route request (RREQ) is flooded, indicating a fresh sequence number of the initiator, the last known sequence number of the destination, and a hop-count field which is initially set to zero. Every node that receives the RREQ

1) updates the sequence number of the source, and adds an RR (for the source) to a table of RRs.
2) if the node does not have a path to the requested destination it forwards the RREQ after incrementing the hop count field by one;
3) if the node has a fresh enough path to the destination (or if the node is the destination itself), it responds by unicasting a route response (RREP) towards the source by sending the RREP to the neighbor from which it received the RREQ.

In the case of RREP by an intermediate node the hop count in RREP is set to the stored value, and in the case of RREP by the destination, it is initialized to zero. Every node receiving the RREP adds an RR for the destination, increments the hop count, and unicasts the packet towards the source node.

Every entry in the routing table has a validity time after which it cannot be used. However, due to the mobility, the information in the RRs can become invalid even before the expiry period. AODV handles such premature expiry using route error (RERR) messages.

DSR also employs a similar RREQ, RREP and RERR packets. The difference is that while in AODV intermediate nodes (that forward the RREQ) increment the hop-count, in DSR every intermediate node inserts its identity. Thus, compared to AODV, DSR provides some additional topology information. The primary disadvantage of DSR is the additional bandwidth overhead for RREQ and RREP packets (which indicate the entire path instead of a single field - hop count). The advantage accrued from the knowledge of the entire path is that multiple paths between the source and destination can be established.

As originally proposed, both AODV and DSR simply ignore the possibility of malicious nodes which intend to disrupt the MANET. Several secure routing protocols have been proposed which add cryptographic authentication of routing data. Cryptographic authentication strategies are primarily intended to ensure that some illegal modifications to the routing data can be identified. Specifically, in secure AODV protocols a specific aim of cryptographic authentication is to ensure that illegal modifications to hop-count values can be detected. The main aim of most secure DSR protocols is to permit detection of illegal insertions or deletions of nodes in the path indicated in RREQ/RREP packets.

B. Trusted Computing Base

The trusted computing base (TCB) of a system is "a small amount of software and hardware we rely on, and that we distinguish from a much larger amount that can misbehave without affecting security" [3].

As an example consider a generic communication system, where an important assurance sought is the ability to verify that a message sent from one entity to an another cannot be modified in transit by intermediaries. To realize such an assurance we typically rely on a TCB which includes a certificate authority (CA), who (we assume) does due diligence before signing public key certificates, and ensures that it's private key is well protected. We also rely on the assumption that cryptographic algorithms like RSA, DSA, AES, SHA-1 etc., are unbreakable. When one receives a message authenticated using the secrets belonging to an entity A, it is assumed that the message is from A, as it is implicitly assumed that the secrets of A are privy only to A.

As a more concrete example, the widely used web-security protocol, SSL, leverages such a TCB to provide an assurance that data sent by clients will be privy only to SSL servers, However, when a client sends some sensitive information (like a credit-card number) to a server over an SSL connection, it only ensures that the information remains private till it reaches the server. There is no assurance that such information cannot be abused *after* it reaches the server, by entities who have unfettered access to the server. Thus, in many practical scenarios, the limited TCB which caters only for cryptographic authentication, is not sufficient as a basis for realizing important assurances.

1) Trustworthy Computing Modules: The most common approach to expand the TCB is by employing trustworthy computing modules which provide some "specialized" TCB functions, performed inside trustworthy boundaries.

In the trustworthy computing group (TCG) model [4] for realizing trusted platforms a trustworthy platform module (TPM) performs several specialized *fixed* functions to provide i) the ability for remote parties to verify that the platform equipped with the TPM is in an "acceptable state" - that only authorized software has been loaded and executed by the CPU (even though the TPM does not have direct control over the

CPU); and ii) the ability to provide secrets to the TPM, bound to some platform states, which will be released by the TPM only when the platform is in that specific state.

Unlike TPMs, the IBM 4758 [5] trustworthy computing module sports a general purpose processor inside a protected boundary, running a specialized operating system, and can execute application code unmolested *inside* the trusted boundary. The rich set of programmable functions that can be executed inside the boundary can provide a rich TCB that can be leveraged to realize assurances that may not be possible otherwise.

Unlike inexpensive TPM chips (a few dollars) with fixed functionality, the programmable TCB offered by IBM module comes at a substantially higher cost (a few thousand dollars). It is for this reason, that in this paper we seek a set of fixed functionality suitable for securing MANETs. We deliberately impose some restrictions on such fixed functionality to ensure that TMMs which offer such functionality can be easily verified, will consume negligible power, and thus can be simultaneously trustworthy and inexpensive to realize.

TMMs that offer the TCB for securing MANETs will demand substantially lower complexity compared to even inexpensive TPM chips. Unlike TPMs which offer a set of about 120 fixed functions, TMMs will offer 3 such functions. Furthermore, unlike TPM chips, TMMs will not require to perform asymmetric cryptographic computations. TMMs will merely perform fixed sequences of logical and cryptographic hash operations.

III. TCB for MANET Nodes

Every mobile computer capable of taking part in any MANET is assumed to be equipped with a TMM. Every TMM is preloaded with a symmetric secret provided by a trusted authority. In practice, the TMM may be a SIM card issued by the trusted authority. The SIM (subscriber index module) card can be plugged into an untrusted mobile computer. Only such computers equipped with the TMM become eligible to participate in MANET subnets, adhering to rules promulgated by the trusted authority.

Any two TMMs A and B (housed in nodes A and B respectively) can use their respective secrets K_A and K_B to independently compute a pairwise secret K_{AB}. Any subset of nodes belonging to the network can come together to create a temporary MANET subnet. While operating in a MANET subnet, a mobile computer uses the interfaces exposed by its TMM to submit some values to the TMM and receive time-stamped message authentication codes (MACs).

When a TMM A is provided some values along with a MAC computed using a secret K_{AB}, and if the time-stamp t_b indicated by B is "close" to the current time t_a (according to TMM A), the TMM A recognizes B is a neighbor. Every TMM maintains a list of neighbors in a neighbor table. TMMs will only accept authenticated routing information from neighbors, and in response, subject to some rules, prepare MACs for messages that are verifiable only by such neighbors.

By maintaining a table of neighbors, and by performing simple and fixed sequences of logical and hash function operations, TMMs ensure adherence to the rules that govern a MANET protocol. More specifically, the assurances realized using the proposed approach is only based on the assumptions that

1) the secret stored inside the TMM cannot be exposed, and
2) the simple and fixed functionality of the TMM cannot be modified

More specifically, the nodes themselves, and the users in control of the nodes are not trusted.

A. High Level Architecture of TMMs

Internally, every TMM consists of

1) a secure hash function $h()$ (say SHA-1);
2) a protected battery-backed RAM (BBRAM) for storing one or a few symmetric secrets - let the secret assigned to the TMM with identity A be K_A;
3) limited RAM for storing a "neighbor table" ($\mathbf{T}$) and some fixed parameters (Δ, δ_n, MAX), and a dynamic sequence number q (TMM A stores its current sequence number q_a);
4) I/O registers;
5) a clock-tick counter: the clock-tick-counters of all TMMs are assumed to have some extent of time synchronization (for example, within a few tenths of a second) to enable them to agree on the expiry time of routing records); and
6) hardwired control logic which drive a sequence of operations consisting of logical and hash operations.

1) Pairwise Secrets Between TMMs: Several key distribution schemes for facilitating pairwise secrets between trustworthy modules have been proposed in the recent past. For scenarios involving trustworthy modules there are compelling reasons to reduce the computational overhead inside the module for the operations performed using protected secrets. For the scheme in [6] each module will be required to store a few tens of keys and perform a few tens of block cipher operations for computing any pairwise secret. For the scheme in [7] each module will need to store a single secret and perform a single block-cipher operation.

While both schemes support asynchronous induction of nodes, the former [6] can support unlimited network sizes; the latter [7] imposes a soft limit on the maximum number of nodes (for example, not much more than a few tens of millions). In this paper we assume that the scheme in [7] is used for facilitating pairwise secrets. Specifically the pairwise secret K_{AB} between A and B is computed by A as

$$K_{AB} = h(K_A \parallel B) \oplus P_{AB}. \qquad (1)$$

where P_{AB} is a pairwise public value. The TMMs do not have to worry about the integrity of the public values. The public values are maintained by nodes. Modifications to the public values cannot result in exposure of the secrets.

2) TMM Interfaces: Mobile nodes (which are untrusted) communicate with their TMM (which is trusted) by writing a few values into the input registers of the TMM, and reading the outputs from the TMMs output registers. Values written into the input registers are typically fields like identities of nodes, MACs, and a clock-tick value. The TMM computes pairwise secrets, and employs such values along with other values provided to the TMM, and some internal values (like its clock-tick), to verify and/or compute MACs. Typically the output of a TMM consists of a few MACs, and a clock-tick value (a time-stamp) necessary for verifying the generated MACs. Such operations performed by the TMM to map inputs to outputs are controlled by the control logic inside the TMM, which essentially makes repeated use of the PRF.

More specifically the TMMs offer three simple interfaces to nodes which house TMMs:

1) UpdateNT()
2) RelayRR ().
3) RelayData ()

Some of such important rules enforced by the TMM A are

1) Every node will increment the hop-count by exactly 1.
2) Routing records will be honored only if authenticated by TMMs associated with neighbors with bidirectional links.
3) Routing records will be authenticated by TMMs only to TMMs associated with bidirectional neighbors
4) Creating a RERR message claiming that a neighbor X is unreachable will result in neighbor X becoming unreachable.
5) Data packets will be afforded end to end protection, and that they cannot be redirected through paths longer that the path specified by the source.

B. Notations

In the rest of this paper we shall use the following notations to describe the functionality of a TMM assigned an identity A.

$h()$: a hash function like SHA-1.

K_A: A TMM with identity A stores a symmetric secret K_A in the protected BBRAM.

t_a: the current clock-tick count of node A.

K_{AB}: A pairwise secret between TMM A and a TMM B, computed as $h(K_A \parallel B) \oplus P_{AB}$ where P_{AB} is a pairwise public value (the public value is provided to the TMM by the node housing the TMM).

$\mathbf{T}$: The neighbor table $\mathbf{T}$ has one row for each neighbor, with four columns per-row. The four values in the row corresponding to a neighbor P are

1) the identity P,
2) the most-recent authenticated time-stamp t'_p from P;
3) the pairwise key K_{AP} shared between A and P, and
4) a value $l_{ap} = \{0, 1, 2\}$ indicating the status of the link to node id_j. Status $l_{ap} = 1$ indicates a verified link to P; $l_{ap} = 2$ indicates a verified bidirectional link; $l_{ap} = 0$ indicates an unverified link.

Δ: a constant indicating the lifetime of routing records.

δ_n: a constant indicating a time after which a link with status 1 or 2 is set to status 0 (if the neighbor has not been heard from for a duration δ_n).

INF: a constant representing infinite (or unknown) hop-count.

q_a: the current sequence number of A;

Routing record $\mathbf{R}_D = D \parallel q_d \parallel n_d \parallel \tau_d \parallel a_d \parallel \nu$: A routing record for a destination D containing i) the identity D, ii) a sequence number q_d; iii) the number of hops n_d (to D); iv) time of expiry τ_d of routing data $\mathbf{R}_D$; v) an auxiliary value a_d which can be a one way function of some additional information regrading the route to D, and vi) a flag ν. The flag ν is zero for regular routing records. If the flag ν is set to one this indicates a recent loss of a link to a neighbor who had earlier provided a record for D with $\nu = 0$. Specifically, for DSR and AODV the flag $\nu = 1$ for route error (RERR) packets.

h_r: Hash of an RR. For example, the hash of the RR for D is $h_r = h(D \parallel q_d \parallel n_d \parallel \tau_d \parallel a_d \parallel \nu)$.

Authentication Record $\mathbf{A}_p = P \parallel h_r \parallel t_p \parallel \mu_{pa} \parallel \alpha_{pa}$: Authentication information for a routing record (for example, $\mathbf{R}_D$) provided by a neighbor P. The value h_r is the hash of a routing record (like $\mathbf{R}_D$). P is a neighbor of A which provided the routing record. The value t_p is the time according to P at which the MAC μ_{pa} was computed, and α_{pa} is a one-bit flag (where 1 represents an acknowledgement). The MAC μ_{pa} is computed as

$$\mu_{pa} = h(h_r \parallel t_p \parallel \alpha_{pa} \parallel K_{PA}). \tag{2}$$

C. *UpdateNT*$(\mathbf{A}, \xi)$

Using this interface node A submits to its TMM A a authentication record $\mathbf{A} = P \parallel h_r \parallel t_p \parallel \mu_{pa} \parallel \alpha_{pa}$, along with a value ξ which specifies the nature of the update required to the neighbor table.

1) $\xi = 0$: If $\xi - 0$ the values $\mathbf{A}$ are authentication parameters for a routing record from a neighbor P. Recall that the entry for P in the neighbor table $\mathbf{T}$ (of TMM A) is of the form $(P, t'_p, K_{AP}, l_{ap})$. The TMM verifies the MAC μ_{pa} using K_{AP} and updates the time-stamp field corresponding to P to t_p (if $t'_p < t_p$).

If the status of P is not 2, viz., $l_{ap} \neq 2$ and if $\alpha_{pa} = 0$ then l_{ap} status is set to 1; if $l_{ap} \neq 2$ and if $\alpha_{pa} = 1$ (the received message is an *acknowledgement*) the TMM A assumes that a bidirectional path exists to P and sets $l_{ap} = 2$.

Irrespective of the value of l_{ap} and h_r, the TMM outputs a time-stamped MAC (with time t_a)

$$\mu_{ap} = h(h_r \parallel t_a \parallel \alpha_{ap} = 1 \parallel K_{AP}), \tag{3}$$

as an acknowledgement to TMM P which provided the authentication record $\mathbf{A}$.

If $l_{ap} = 2$, $h_r \neq 0$, and $\alpha_{pa} = 0$, the TMM A additionally outputs a *self-MAC*

$$\theta = h(h_r \parallel P \parallel K_A). \tag{4}$$

This self-MAC θ can be submitted to the TMM A at any later time to prove that a value h_r was authenticated by P.

2) Other Values of ξ: If type $\xi = 1$ and $\mathbf{A} = \{Y \parallel h_r \parallel 0 \parallel 0 \parallel 0\}$ the TMM interprets this as a request to add an entry for a node Y in its neighbor table with status l_{ay} set to 0 and time-stamp $t'_y = 0$. The value h_r is interpreted as the public value P_{AY} associated with Y.

If type $\xi = 2$ and $\mathbf{A} = \{Y \parallel 0 \parallel 0 \parallel 0 \parallel 0\}$ an entry for Y is removed from the neighbor table (if it exists).

If type $\xi = 3$ and if $\mathbf{A} = \{X \parallel 0 \parallel 0 \parallel 0 \parallel 0\}$, the TMM interprets this as a request to output a MAC for the value 0 verifiable by every node in $\mathbf{T}$ of A (irrespective of the link-status of the node). A MAC for a neighbor B is computed as

$$\mu_{ab} = h(0 \parallel t_a \parallel 0 \parallel K_{AB}). \tag{5}$$

Such MACs accompany HELLO packets sent by A.

D. *RelayRR*($\mathbf{R}, P, \theta, \phi$)

The inputs to RelayRR() function of TMM A are are i) a RR $\mathbf{R}$ with hash h_r; ii) the identity of the provider P; iii) a self-MAC $\theta = h(h_r \parallel P \parallel K_A)$ verifiable by TMM A; and iv) a value ϕ which specifies the protocol (for example, $\phi = 0$ for AODV and $\phi = 1$ for DSR).

Let us assume that the RR is for a destination D, of the form $\mathbf{R}_D = D \parallel q_d \parallel n_d \parallel \tau_d \parallel a_d \parallel \nu$. The TMM of A verifies the self-MAC θ, and checks that the time of expiry τ_d is greater than the current time t_a.

If the neighbor table $\mathbf{T}$ of TMM A does *not* have an entry for P (or if $l_{ap} \neq 2$), the TMM sets $\mathbf{R}'_D = D \parallel q_d \parallel n_d = INF \parallel \tau_d \parallel a_d = h(A) \parallel \nu = 1$, and computes a MAC for every neighbor with status 2.

On the other hand, if P is listed as a bidirectional the TMM A sets $\mathbf{R}'_D = D \parallel q_d \parallel n'_d \parallel \tau_d \parallel a'_d \parallel \phi \parallel \nu$, where

1) $a'_d = h(a_d \parallel A)$ if $\phi = 1$ (for DSR), and $a'_d = a_d$ if $\phi = 0$ (for AODV)
2) $n'_d = n_d + 1$ if $n_d \neq INF$,

and outputs a MAC (for $h'_r = h(\mathbf{R}'_D)$) for all neighbors with status 2 (except the provider P).

If the identity of the provider is the node itself ($P = A$) a fresh RR for A is created. In the unauthenticated RR $\mathbf{R}_A$ submitted to the TMM, the node A is allowed to specify *any* auxiliary value a_a. The TMM updates its current sequence number q_a by 1, sets $\mathbf{R}_A = A \parallel q_a \parallel n_a = 0 \parallel \tau_a = t_a + \Delta \parallel a_a \parallel \nu = 0$.

Most often such an RR initiated A accompanies a RREQ from A. The node A may specify other parameters of the request (like intended destination, maximum hop count and last known sequence number of the destination etc.) by choosing an appropriate a_a (for example, as a hash of all immutable RREQ fields).

E. *RelayData*($S, d_h, U, \theta_u, \mathbf{R}, P, \theta_p$)

Consider a scenario where a data packet with hash d_h is relayed from originator S to destination D and that the packet is received by the intermediate node A from an immediate upstream node U. Also assume that P is immediate downstream node (the next hop in the path to D).

The value μ_u is a MAC computed by U as

$$\mu_{ua} = h(h_d = h(S \parallel D \parallel d_h) \parallel t_u \parallel \alpha_{ua} \parallel K_{UA}) \tag{6}$$

If A is the creator of the data packet with hash d_h then $S = U = A$ and $\mu_u = 0$. In this case the TMM does not verify the integrity of the MAC of the previous hop.

To forward a data packet the node A is required to demonstrate to its TMM A a valid record $\mathbf{R}_D$ for destination D, provided by the next hop P. Recall that θ_p is the self-MAC for the record $\mathbf{R}_D$ provided by P. The TMM outputs a MAC

$$\mu_{ap} = h(h_d = h(S \parallel D \parallel d_h) \parallel t_a \parallel \alpha_{ap} \parallel K_{AP}) \tag{7}$$

IV. Comparison with Other Secure Protocols

We shall now see how three functions UpdateNT(), RelayRR() and RelayData() can be used to secure on-demand MANET protocols like AODV and DSR. Following this, we compare the assurances realized by the proposed approach with those realized by popular secure extensions to DSR and AODV.

A. *Realizing AODV and DSR*

When TMM enabled nodes are used in a MANET subnet employing AODV/DSR the only difference is that along with plain routing packets, every node sends some additional values - a clock-tick value and some MACs. RREQ and RERR packets are accompanied by one MAC for every bidirectional neighbor. RREP and data packets are accompanied by one MAC for the next hop.

Consider a scenario where a node A enters a subnet and recognizes the presence of nodes B, C and D within its range. At this point, while node A recognizes its neighbors, the TMM of A does not. The TMM of A recognizes a node B as a neighbor only if a time-stamped and authenticated MAC (authenticated using secret K_{AB}) is provided to the TMM.

Node A uses the UpdateNT() function to add nodes B, C and D to its neighbor table with status set to 0. As the pairwise key for B, C and D is available after this step, now A can request its TMM to compute MACs corresponding to a time-stamped HELLO packet (with $h_r = 0$) verifiable by its neighbors B, C and D. On receipt of the HELLO packet, B submits the time stamped MAC to its UpdateNT() resulting in the addition of A as a neighbor with status $l_{ba} = 1$. When an acknowledgement is sent from B with an authenticated time-stamp, this can be submitted by A to its TMM resulting in the addition of B to the neighbor table of A with status $l_{ab} = 2$. An acknowledgement for this packet from B will result in the $l_{ba} = 2$ in B's table.

In the same way, a list of neighbors are maintained in the neighbor table of all TMMs, characterized by the link status and the latest time-stamp. Periodically, nodes may send supercilious HELLO packets to ensure that the TMMs of neighboring nodes recognize their presence. The MACs necessary to authenticate HELLO packets are also obtained by using the interface UpdateNT().

Every routing packet can be seen as an information record about a specific node in the network. A RREQ provides information about the source (node that created the RREQ), while RREP packets provide information about the destination. TMMs do not differentiate between RREQ and RREP packets. Most often, the job of a node is to simply relay RREQs and RREP packets after updating the hop count (or hop-count and an auxiliary value in DSR). TMMs ensure that routing information will be honored only from neighbors with status 2, and will be sent only to neighbors with status 2.

To create a RREQ for a destination T the source A constructs the regular RREQ packet (indicating source, source sequence number, destination, last known destination sequence number, etc) and computes the hash of the all immutable contents (fields that do not change as the RREQ propagates) to derive the auxiliary value a_a that will accompany the RREQ packet. In both DSR and AODV the TMM will ensure that the sequence number of the source A is incremented every time the interface RelayRR() is used for creating an RREQ. (by setting the provider identity to own identity).

In both AODV and DSR, an intermediate node B extracts the values $\mathbf{R}$ and $\mathbf{A}$ from the RREQ/RREP/RERR packet received from a neighbor (say P). While in AODV the auxiliary value a in $\mathbf{R}$ is extracted using only the immutable fields, in DSR the path of the packet till B is also used to extract the auxiliary value. The value h_r in $\mathbf{A}$ is then computed by hashing $\mathbf{R}$. The value h_r and the corresponding authentication record $\mathbf{A}$ are then submitted to the UpdateNT() interface where The TMM updates the neighbor table, and issues a self-MAC θ to the effect that "a value h_r was received from a neighbor P"

Following this, the RR $\mathbf{R}$ and the self-MAC θ for the RR are cached by the node. At any later time the RR $\mathbf{R}$ can be submitted to the TMMs RelayRR() interface (as long as the time τ specified in the RR has not expired). It is only the RelayRR() function which performs some simple protocol specific steps. For AODV (if the value $\phi = 0$ in the RelayRR() interface) TMM B simply increments the hop count value n; on the other hand (if $\phi = 1$ for DSR), TMM B increments the hop count value *and* extends the auxiliary value a to $h(a \parallel B$.

Thus, if $\phi = 0$ (AODV) the auxiliary value is relayed unchanged. IF $\phi = 1$ (DSR) the auxiliary value is extended to cryptographically bind the identity of the relaying node. For an RREQ packet that has traversed through nodes P, Q, R and reached a node S, the value a in $\mathbf{R}$ received by S is $h(h(h(P \parallel h(I)) \parallel Q) \parallel R)$. Where I is all the immutable information included by the initiator A.

In both AODV and DSR an intermediate node with a fresh enough path can respond to an RREQ for D by initiating an RREP - by submitting the valid $\mathbf{R}_d$ that it previously received. Assume that Q had initially received a valid $\mathbf{R}_d$ for destination D from P, and later received a RREQ for D from a different neighbor N. Node Q can now submit the $\mathbf{R}_d$ it previously received from P along with the self-MAC θ that authenticates the $\mathbf{R}_d$, and convey this information to neighbor N (as long as N is a neighbor with link-status 2).

In instances where the destination node, say D, wants to create a RREP, RelayRR() is invoked to create MACs that authenticate the destination node itself (sets the provider as D). The auxiliary field a submitted to this function call is hash of all the immutable fields included in a RREP of the corresponding protocol (either AODV or DSR). For DSR a also represents the source path included in the packet.

While generating RREPs the TMM would output MACs which authenticate the packet to all the listed bidirectional neighbors. As RREP is unicast a node would only include the MAC that is verifiable by the intended next-hop - the other MACs can be ignored by the node.

A node creates RERR packets to notify prematurely expired RRs. In our scheme a node can create an authenticated RERR for a destination D (by setting $\nu = 1$ in a routing record for D) only by i) submitting an authenticated routing record for D ($\mathbf{R}_d$ along with a self-MAC), *and* demonstrating that ii) the provider of $\mathbf{R}_d$ is no longer listed a bidirectional neighbor in $\mathbf{T}$. A TMM then then creates an RERR by setting $n = INF$ and $\nu = 1$, and authenticates it to all the listed bidirectional neighbors. However, relaying an RERR packet is no different from relaying a regular RREQ or RREP packet (with the exception that n_d is already INF and is not incremented, and $\nu = 1$ in the record).

Finally, to forward data packets a node has to submit the received data packet from a neighbor U along with a previously received authenticated RR $\mathbf{R}$ for the destination of the data packet (say RR received from a neighbor P). The TMM verifies the submitted $\mathbf{R}$ and authenticates the received data packet to the next-hop (P - provider of the RR). In this section we shall first see how compare the security assurances offered by our scheme with popular secure extensions of AODV and DSR.

B. Some Secure MANET Protocols

In the Secure AODV (SAODV) protocol [8] every node has a public-private key pair with a certified public key. Digital signatures are used to authenticate immutable fields in RREQ, RREP and RERR messages specified by the source of the packet (an end-point). The immutable field includes a commitment to a hash chain of length $x_0 \cdots x_n$, (where $x_i = h(x_{i-1})$, and $h()$ is a cryptographic hash function like SHA-1) where n is the maximum length of the path. The RREQ from the source (indicating hop-count 0) is accompanied by a value x_0. Nodes at the first hop are required increment the hop-count to 1 and propagate the RREQ along with the value $x_1 = h(x_0)$, and so on.

In order to secure route responses by intermediate nodes SAODV employs double signature extensions for RREQ and RREP packets. The RREPs generated by an intermediate node includes the signature of the destination to validate the immutable fields, and also a signature of the intermediate node to authenticate the new validity time.

In the SAODV-2 protocol [9] only symmetric cryptography is used. SAODV-2 argues that schemes for establishment of pairwise secrets between two nodes demand substantially lower computational and bandwidth overhead. Further, SAODV-2 uses two hop authentication to thwart illegal hop changes. In two-hop authentication a node (say A) appends a MAC which is verifiable by its two-hop neighbors (say C). When C receives this packet via a B (neighbor of A) it can compare the hop count announced by B to the one included by A and identify illegal modifications.

1) Secure DSR: A well known secure extension of DSR is Ariadne [10] which uses the TESLA broadcast authentication protocol [11] for authenticating network packets. Every intermediate node forwarding an RREQ appends a TESLA MAC which can be verified at the end of the reverse path. As the authentication appended by every node will be verified by the RREQ source, nodes cannot be inserted into the path. To prevent nodes from deleting other nodes in the path a per-hop hashing strategy is used which leverages a shared secret between end-points (the RREQ source and the destination).

In [12] an improved Ariadne (iAriadne) was proposed which mandates every node to maintain a "private logical neighborhood" (PLN) and introduces an additional value to be inserted by every node forwarding the RREQ - an "encrypted up-stream per-hop hash." Unlike Ariadne with TESLA, iAriadne relies on pairwise secrets between nodes.

C. Shortcomings of Current Secure Protocols

Even while demanding substantial overhead, SAODV still leaves an AODV MANET susceptible to a wide range of attacks. Some of the main shortcomings of SAODV (which are addressed in SAODV-2) are

1) lack of mechanisms to authenticate intermediate nodes (intermediate nodes are not required to append any authentication)
2) inability to prevent some misrepresentations of the hop-count: it can only ensure that a node receiving an RREQ/RREP with hop count r cannot relay a hop count less than r. A malicious node can incorrectly (or maliciously) relay the same hop count r or a hop count greater than r, and
3) lack of mechanisms to address one-way links.

While SAODV-2 addresses many of the pitfalls of SAODV, SAODV2 is still susceptible to many attacks. SAODV-2 assumes that nodes will not collude together. Colluding nodes can easily thwart two hop authentication. Secondly, there is no mechanism for regulating creation of RERR packets.

Unlike SAODV (where intermediate nodes are not required to append authentication) in Ariadne authentication of intermediate nodes is mandatory. Unfortunately, the verification of the authentication occurs very late in the RREQ-RREP process. Due to this malicious nodes in the path can send random RREQ packets which even though will not be accepted by end-points, can preempt propagation of genuine RREQs.

The main pitfalls of Ariadne that are addressed by iAriadne are addition of mechanisms for i) link-layer (one-hop) authentication, and ii) preventing abuse of one-way links - both of which are achieved by imposing a private logical neighborhood (PLN). The additional upstream per-hop hash facilitates the RREQ destination to narrow down intermediate noes that engage in active attacks.

1) Collusion: However, the security of all four protocols are based on the assumption that nodes will not collude. In both SAODV and SAODV-3 colluding nodes can ensue that even shorter hop counts can be relayed. In Ariadne and iAriadne colluding nodes can trivially delete nodes from the path.

The primary reason that facilitates easy collusion is that two nodes A and B simply need to share their secrets to do so. Through their ability to send packets impersonating A or B, both A and B can create packets with misleading information, with the false information provided by A consistent with the false information provided by B. Obviously, three such nodes A, B, C sharing their secrets can engineer an even broader variety of attacks.

2) Unregulated RERR Creation: Another pitfall of all protocols is that they do not possess secure for creation of RERR packets. An effective strategy for an attacker to introduce unnecessary additional overhead is to participate faithfully in establishing a path and then send a supercilious RERR packet - mandating a fresh route creation process. For example an attacker C in a path $A \rightarrow B \rightarrow C \rightarrow X$ can simply send a RERR claiming to have lost the link to X (even while the link exists). A selfish node desiring to retain X as a neighbor can still manage to send such a RERR packet by ensuring that the RERR will not be heard by X. This can be achieved easily by exploiting the medium access control protocol. For example, if a collision avoidance protocol is used, C can send the packet as soon as X sends a clear to send (CTS) or a request to send (RTS) packet, thereby ensuring that C's RERR suffers collision at X (but will be received collision-free by other neighbors of C).

D. TMM Based Secure Protocols

In the TMM based approach a node cannot produce an authenticated routing record without submitting an authenticated routing record to its TMM. All that the TMM needs to do is to ensure that the routing record originates from an active neighbor, make appropriate changes to the record (increment hop-count, and extend the auxiliary value if $\phi = 1$), and authenticate the modified record to all neighbors with link-status 2. If a valid record is received from a node which is *not* in the neighbor table (or has status 0 or 1), the modification required is that the flag ν is set to 1 in the routing record and authenticated to all neighbors.

Thus, an RERR packet can be created by a node only if a valid routing record exists and the neighbor which provided the record is no longer a neighbor (indicating recent loss of a link). This makes it impossible for nodes to send RERR packets without actually losing a neighbor. In practice, a node desiring to send an RERR has to remove a current neighbor to send an RERR. The risk of loosing a neighbor (and thereby links to other nodes through that neighbor) can be an effective deterrent against such attacks.

The problem of collusion is addressed simply by shutting out the avenue which facilitates easy collusion in the first place - the ability to easily share each others secrets. By ensuring that secrets used for authentication of packets from node A

are privy only to TMM A and relying on the ability of TMMs to protect secrets we can address the root cause of collusion.

To achieve the required security goals, viz.,

1) ensuring that routing records cannot be modified illegally
2) ensuring that routing records will be accepted only from nodes with tested bidirectional links and will be relayed only to nodes with tested bidirectional links,
3) regulating RERR creation, and
4) addressing collusion based attacks

the nature of operations performed inside the TMM are truly trivial.

A TMM can only make a node to accept authenticated RRs, and ensure that changes to received RRs are made adhering to the protocol. However, the TMM cannot force the node to broadcast a packet as the communication interfaces are not included within trusted boundaries. Hence explicit provisions are required to discourage nodes from maliciously dropping packets. It is for this purpose we provide the node with the ability to remove a node from the neighbor table $\mathbf{T}$ of its TMM. Nodes which identify selfish behavior by neighbors can simply remove the neighbor. Unwarranted dropping of packets can result in a node being disconnected from the network as the neighbors of the node would remove the node from their neighbor list.

In other words, *active* attacks (involving illegal modifications to routing packets) are addressed by TMMs by ensuring that the rules governing the underlying routing protocol cannot be violated. Passive attacks involving selective or selfish participation are addressed by the nodes through their ability to eject nodes from their "logical neighborhood" - the neighborhood as seen by the TMM of the node.

V. Conclusions

For any security solution we begin with the assumption that some components/algorithms are trusted, and amplify this trust to realize the desired assurances. The obvious question is then "what is this minimal trusted computing base (TCB) to realize the desired assurances?" It is this question that led to the proposed TMM based approach.

The need for trusted boundaries in which co-operative routing tasks are carried out has been addressed by some researchers. In [13] the authors include the wireless transceiver inside the trust boundary. In [14] the trusted computing module has complex features built into the wireless driver (executed within the confines of the trusted module) to verify the integrity of wireless transceiver. Arguably, bringing the wireless transceiver within the scope of the protected boundary as in [13] or including complex features in wireless software drivers (executed within the trusted boundary) as in [14] implies high cost for practical realization of such trust modules.

In [15] explicit consideration is given to the need for lowering the complexity of tasks to be performed inside the trusted boundary. The scheme employs "nuglets of currency" protected by smart-cards to promote faithful forwarding of packets. Nevertheless [15] still assumes that the trusted computing modules should be capable of performing asymmetric cryptographic computations, which raises the bar for the capabilities and the cost of such modules. More recently Gaines et al [16] have proposed a generic dual-agent approach to MANETs where some desired characteristics of a trustworthy network agent (like low computational and storage requirements) are enumerated.

One of the primary motivations for the proposed approach is to minimize the complexity of operations performed inside the TMM to the extent feasible, but yet effectively curtail the freedom of an attacker. Lower the complexity of the TMM, lower the cost, and higher is the extent of trust one can place on the immutability of the TCB. We believe that the current approach can be easily extended to support other routing protocols like DSDV, TORA, etc., by small modifications to the RelayRR() function. Specific changes required for this purpose is our current research focus.

References

[1] C. Perkins, E.Royer, S. Das "Ad hoc On-demand Distance Vector (AODV) Routing, Internet Draft, draft-ietf-manet-aodv-11.txt, Aug 2002. The 6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2002), 2002.

[2] P. Johanson, D. Maltz, "Dynamic source routing in ad hoc wireless networks," Mobile Computing, Kluwer Publishing Company, 1996, ch. 5, pp. 153-181.

[3] B. Lampson, M. Abadi, M. Burrows, E. Wobber, "Authentication in Distributed Systems: Theory and Practice," ACM Transactions on Computer Systems, 1992.

[4] TCG Specification: Architecture Overview, Specification Revision 1.4, 2nd August 2007.

[5] S.W. Smith, S. Weingart, "Building a High-Performance Programmable Secure Coprocessor," IBM Technical Report RC21102, Feb 1998.

[6] M. Ramkumar, "The Subset Keys and Identity Tickets (SKIT) Key Distribution Scheme," IEEE Transactions on Information Forensics and Security, **5**(1), pp 39–51, March 2010.

[7] M. Ramkumar, "On the scalability of a "non-scalable" key distribution scheme," IEEE SPAWN, Newport Beach, CA, June 2008.

[8] M.G.Zapata, N.Asokan, "Securing Ad hoc routing protocols," WISE-02, Atlanta, Georgia, 2002.

[9] K.A. Sivakumar, M. Ramkumar, "Safeguarding Mutable Fields in the AODV Route Discovery Process," the Sixteenth IEEE ICCCN-07, Honolulu, HI, Aug 2007.

[10] Y-C Hu ,A Perrig,. D B.Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," Journal of Wireless Networks, **11** pp 11–28, 2005.

[11] A. Perrig, R. Canetti, D. Song, D. Tygar, "Efficient and Secure Source Authentication for Multicast," in Network and Distributed System Security Symposium, NDSS '01, Feb. 2001.

[12] K.A. Sivakumar, M. Ramkumar, "Improving the Resilience of Ariadne," IEEE SPAWN 2008, Newport Beach, CA, June 2008.

[13] J-H. Song, V. Wong, V. Leung, "Secure Routing with Tamper Resistant Module for Mobile Ad Hoc Networks," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 7, no. 3, ACM Press, New York, Jul. 2003.

[14] M. Jarrett and P. Ward, "Trusted Computing for Protecting Ad-hoc Routing," Proceedings of the 4th Annual Communication Networks and Services Research Conference, IEEE Computer Society, May 2006.

[15] J-P. Hubaux, L Buttyan, S. Capkun, "Quest for Security in Mobile Ad Hoc Networks," Proceedings of the ACM MOBIHOC 2001.

[16] B. Gaines, M. Ramkumar, "A Framework for Dual Agent Routing Protocols for MANETs," IEEE Globecom 2008, New Orleans, LA, Nov 2008.