TECHNICAL RESEARCH REPORT

Bootstrapping Security Associations for Routing in Mobile Ad-Hoc Networks

by R.B.Bobba, L.Eschenauer, V.D.Gligor, W.Arbaugh

TR 2002-44

Bootstrapping Security Associations for Routing in Mobile Ad-Hoc Networks

Rakesh Babu Bobba, Laurent Eschenauer, Virgil Gligor, and William Arbaugh
University of Maryland
College Park, Maryland 20742
{bobba, laurent, gligor}@eng.umd.edu, waa@cs.umd.edu

Abstract— To date, most solutions proposed for secure routing in mobile, ad-hoc networks (MANETs) assume that secure associations between pairs of nodes can be established on-line; e.g., by a trusted third party, by distributed trust establishment. However, establishing such security associations, with or without trusted third parties, requires reliance on routing layer security. In this paper, we eliminate this apparent cyclic dependency between security services and secure routing in MANETs and show how to bootstrap security for the routing layer. We use the notion of statistically unique and cryptographically verifiable (SUCV) identifiers to implement a secure binding between IP addresses and keys that is independent of any trusted security service. We illustrate our solution with the Dynamic Source Routing (DSR) protocol and argue that the solution is applicable to other protocols such as SEAD and Ariadne. We evaluate the cost of the DSR solution with simulations over *ns-2* and present some preliminary results.

***Index Terms*—mobile ad-hoc networks, routing, security, DSR**

I. Introduction

In contrast to the Internet where end users do not perform routing functions, mobile ad-hoc networks (MANETs) require end-user nodes to perform packet routing. In these networks, a pre-deployed, dedicated network fabric does not exist that routes packets and protects route integrity. Furthermore, MANET nodes are highly mobile and, consequently, the network topology changes frequently. Hence, every node must be able to maintain network connectivity, not just perform packet routing.

Early protocols that performed routing in MANETs [2,7,13,14,15] assumed that all nodes were trusted; i.e., none of the nodes deliberately disrupted the routing protocol. More recently, several protocols were proposed to secure the routing layer from nodes that act maliciously and:

- convince a node that it is on a route to the given destination when it is not.
- lower the cost of a route thereby re-directing traffic to desired nodes;
- fabricate route-maintenance messages for selected routes or links.

These protocols integrate security features within traditional routing protocols, such as DSR [7], AODV[16], DSDV[15], and aim to protect against message modification, fabrication or address spoofing through cryptographic means [3,5,13]. However, all these protocols assume that secure associations between the nodes of the network exist or can be established on-line.[1] Typically, these associations consist of either symmetric keys shared between any two nodes distributed with the help of a trusted key distribution center (KDC), or public-key certificates associated with individual nodes and signed by a trusted certification authority (CA). More recently, a distributed service for establishing trust relations among network nodes from PGP certificates has been proposed that does not rely on a trusted authority infrastructure [6]. Security associations and trust relations among nodes form the basis for building the security features of the routing layer; e.g., message authentication, replay detection.

The assumption of pre-established secure associations may be practical in environments where such associations can be established *off-line* [17]. However, this assumption is less suitable for secure routing in large MANETs where secure associations have to be setup on-demand and *on-line*. In this case, a cyclic dependency arises between security services (e.g., certificate distribution, shared key generation, distributed trust establishment) and routing services since security services require routing layer security themselves (viz., Section II). To break this dependency, security associations must be bootstrapped into the routing layer without reliance on *any* security services.

In this paper, we show how to bootstrap secure associations for the routing protocols of MANETs on-line, without assuming any trusted authorities or distributed trust-establishment services. We rely on the use of *statistically unique and cryptographically*

[1] Other protocols rely on *intrusion-detection* mechanisms to discover and isolate malicious nodes [8], and tacitly assume that the intrusion-detection sensors running in network nodes are somehow protected from the nodes in which they are installed. In the end, these protocols still need to assume the presence of security associations to remove some of the less realistic trust assumptions made.

verifiable (SUCV) identifiers [10] and public-secret key pairs generated by the nodes themselves, in much the same way SUCVs are used in MobileIPv6 (MIPv6) to solve the address "ownership" problem [10,12] and to counter the "bidding down" attack [11] in return routability. We present the bootstrapping solution in the context of the *Dynamic Source Routing* (DSR) protocol and argue that the solution is applicable to other secure routing protocols, such as SEAD [3] and Ariadne [5]. We evaluate the performance of our solution using a ns-2 simulation, and present the preliminary results.

II. Bootstrapping Secure Associations

A. A Cyclic Dependency Problem

Routing and security are separate services in any network. The routing service depends on security services to authenticate the source of a message (i.e., its IP address) and the message content. For example, routing needs to determine that the source address of a route-request, route-reply, or control message was not spoofed, and that the message was not modified by malicious nodes while in transit between a source and a destination. The dependency between secure routing and security services is identified by arrow (1) in Figure 1. Message authentication is typically implemented using either shared symmetric keys or public-private key pairs and requires that the shared or public keys are associated (bound) in a secure way with the nodes' (IP) addresses, *on-line*.

To obtain secure bindings between a node's IP address and key, a node must either reach a trusted-authority node or must establish trust relationships with other nodes without relying on trusted authorities [6]. Even if we assume that the IP address of all nodes are known *a priori*, there is still the need to establish a route from any node to a trusted-authority node or to a peer node that does *not* include malicious nodes. If malicious nodes are present on any of these routes, they may launch denial-of-service attacks and, worse yet, impersonate trusted servers or peer nodes. Secure routing seeks to counter these attacks, typically by detecting the effects of malicious node actions on a route, and by finding alternate routes. Hence, both trusted authorities and distributed trust establishment without trusted authorities depend on secure routing services. This dependency is identified by arrow (2) of Figure 1.

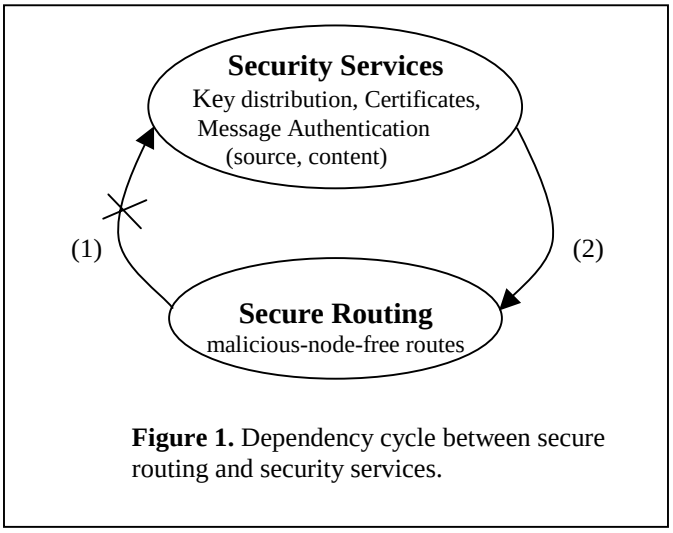

Figure 1. Dependency cycle between secure routing and security services.

In summary, a cyclic dependency arises between security and secure-routing services, which we seek to remove for any secure implementation of routing services.

B. Breaking the Cyclic Dependency

To break the dependency cycle illustrated in Figure 1, we have to remove dependency (1). Removing dependency (2) would be impractical because it would require that the nodes implementing security services be reachable by all other nodes in the network by a fixed set of routes; to implement any routing function for these nodes would defeat the purpose. We remove dependency (1) by using a *secure binding mechanism* for establishing secure node-to-node associations that is *independent* of *both* secure routing and other security services (viz., Figure 2). This mechanism forms the building block for bootstrapping security associations for secure routing protocols.

The idea of a secure binding between an IP address and a key that is independent of any other security

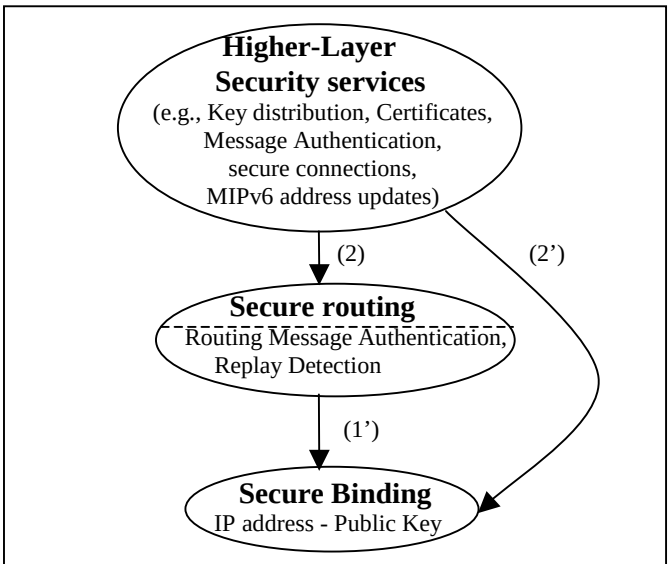

Figure 2. Breaking the cyclic dependency between secure routing and security services.

services is due to O'Shea and Roe [12] and Montenegro and Castelluccia [10]. They used it in the context of securing the control messages of MIPv6. We briefly review the idea of secure <IP address, public key> binding for the sake of completeness and ignore its MIPv6 application specifics.

To generate an IP address that is securely bound to a public key, a node generates a 64-bit pseudo-random value by applying a one-way, collision-resistant hash function to the public key of its (uncertified) public-private key pair. Then, the IP address is generated as the concatenation of a network specific identifier (64 bits in MIP6) and the hash of the public key (64 bits). The binding between this IP address and the public key is secure because it is computationally unfeasible for an attacker (1) to create another <public, private> key pair whose hash generates the same IP address (because of the second pre-image resistance of one-way, collision-resistant hash functions), and (2) to discover the secret key, or create a different one, for a given public key (by definition). Due to the size of the resulting address space, this IP address is also *statistically* unique.

A source node uses the secure binding to authenticate its IP address and the contents of its packets to an arbitrary destination node as follows. The node signs a packet with its private key and sends the signed packet to the destination address together with its public key (and IP address). The destination node verifies that the IP address is securely bound to the public key by computing the hash function of the received public key and comparing the result with the lower 64-bit field of the IP address. (Thus, the IP address "certifies" the validity of the public key thereby preventing an attacker from spoofing the source address.) Then, the destination node authenticates the content of the packet by verifying the signature with the public key.

This authentication scheme is implemented without using any security services. As illustrated in Figure 2, dependency (1) is removed and replaced with dependency (1'), which does not lead to a cyclic dependency. Also, dependency (2') is added to indicate that higher-level security mechanisms, such as those of MIPv6, need to enhance the basic secure binding mechanism (e.g., to allow IP address changes) for different applications. Note that routing-layer authentication is also used to implement simple replay-detection mechanisms for routes (viz., Section III below). Finally, we note that bootstrapping of additional, different security associations for the routing layer becomes unnecessary.

III. SECURING DSR

A. Basic scheme

In this paper, we do not aim to provide a complete description of DSR security and, instead focus only on providing end-to-end security for the basic DSR [7] features without optimizations as the optimizations in DSR make the protocol vulnerable to attacks [13]. Our approach is similar to the one of Papadimitratos and Haas [13], except that we bootstrap the secure association in the protocol itself, by using secure <IP address, public key> bindings. We illustrate the basic protocol using an example topology shown in Figure 3.

DSR is composed of two basic services, namely route discovery and route maintenance. In route discovery any source node S wishing to send a packet to any destination node D, to which it has no cached route, discovers a route to D. In route maintenance, source S detects if the route it is using to D is still valid in the face of topology changes, and, if the route is no longer valid, it discovers a new one.

Route Discovery. When a source node S wants to discover a route to destination D, it initiates route discovery. It constructs a *Route Request* message including the source (S) and destination (D) identifiers, a unique request sequence number, and an (initially empty) list to accumulate the addresses of intermediate nodes forwarding the request to D. Source S digitally signs the source and destination identifiers and the sequence number. It appends the signature and its public key (1024 bits) to the packet and broadcasts it. Each intermediate node receiving the packet appends its address to the node list and rebroadcasts the packet. If an intermediate node finds that its address is already on the node list, it discards the packet.

When the *Route Request* reaches its destination, node

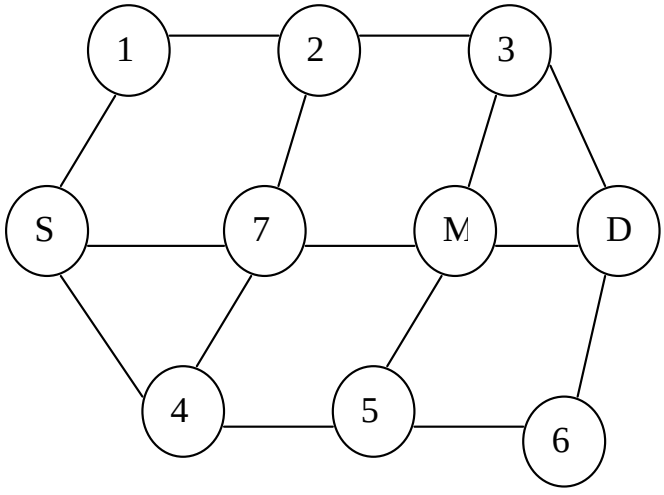

Figure 3. Example topology. S is trying to communicate with D. M is a malicious node.

D authenticates the *Route Request* packet. First, D verifies the validity of the public key by hashing the public key and comparing the result to the lower half of the source S's IP address in the packet. Then, if the comparison indicates a match, D verifies the signature on the packet. If the signature is valid, then D checks to see if the sequence number in *Route Request* is greater than the last sequence number it has seen from S, if true D constructs a *Route Reply* packet. It extracts the accumulated path in the *Route Request*, includes a copy of it in the *Route Reply* packet, and digitally signs the destination and source addresses, the sequence number, and the accumulated path. It then appends its public key and the signature to the packet, and source routes the packet on the reverse of this accumulated path.

When source node S receives the *Route Reply* packet, it first authenticates the (source and content of the) packet and verifies the reply's validity. To authenticate the packet, S verifies the public key of destination node D by hashing it and comparing it against the lower half of the address of D. If the comparison yields a match, then S verifies the D's signature on the packet. If the signature verification passes, D proceeds to verify the validity of the reply. First, S checks if it has a pending *Route Request* to node D with the sequence number returned by the *Route Reply* packet. If it has one, then S extracts the node list out of the packet and compares it against the reverse of source route in the packet header. If this comparison yields a match, the reply is valid and S caches this route. If any of the above checks fail, S discards the reply packet. Figure 4 shows the flow of messages for route discovery.

Route Maintenance. When sending a packet on an established route, source S includes the complete sequence of nodes through which the packet should travel. Each node along the path forwards the packet to the next node indicated in the path. If any node fails to forward it due to link failure or any other reason (detected by the link layer, in our case 802.11 MAC protocol), it signs the packet, appends its public-key, it source routes a *Route Error* to the original source of the packet (S) on the reverse of the source route contained in the data packet, thereby identifying the broken link from itself to the next node.

When source S receives a *Route Error* packet, it first authenticates it by checking the whether the hash of the public key matches the lower half of the IP node that signaled the error, and whether the signature is valid. Then S checks the validity of the *Route Error* packet. S checks whether the address of the node that signaled the error is at the head of broken link in the route. If any of the checks fail, S discards the error packet. Then, S removes the broken link from its route cache. For subsequent packets or the retransmission of this packet, source S may use any other route to that destination available in its cache or may initiate new route discovery.

State maintenance. Note that intermediate nodes do not maintain state in our approach. That is, nodes store neither the <sequence number, source S> pair as in DSR

S→ * (Rq,S,D,#,(empty list)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

1→ * (Rq,S,D,#,(1)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

2→ * (Rq,S,D,#,(1,2)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

3→ * (Rq,S,D,#,(1,2,3)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

D→ 3 (Rp,SR(3,2,1),S,D,#,(1,2,3)) $\{ SR(3,2,1),S,D,\#,(1,2,3)\}_{SK\text{-}D}$ (PK-D)

3→ 2 (Rp,SR(3,2,1),S,D,#,(1,2,3)) $\{SR(3,2,1),S,D,\#,(1,2,3)\}_{SK\text{-}D}$ (PK-D)

2→1 (Rp,SR(3,2,1),S,D,#,(1,2,3)) $\{ SR(3,2,1),S,D,\#,(1,2,3)\}_{SK\text{-}D}$ (PK-D)

1→ S (Rp,SR(3,2,1),S,D,#,(1,2,3)) $\{ SR(3,2,1),S,D,\#,(1,2,3)\}_{SK\text{-}D}$ (PK-D)

Figure 4. Message flow for the discovery of route {1,2,3} between source S and destination D of Figure 1. *Legend*: ***** indicates broadcast address. Rq and Rp are request and reply tags. SR(x,y,z) indicates source route x to y to z. SK-X and PK-X indicate secret key and public key of X respectively. $\{abc\}_{SK\text{-}X}$ indicates signature of X on abc. # indicates sequence number.

S→ * (Rq,S,D,#,(empty list)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

1→ * (Rq,S,D,#,(1)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

2→ * (Rq,S,D,#,(1,2)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

3→ * (Rq,S,D,#,(1,2,3)) $\{S,D,\#\}_{SK\text{-}S}$ (PK-S)

D→ 3 (Rp, SR(3,2,1),S,D,#,(1,2,3)) $(K_{SD})_{PK\text{-}S}$ { SR(3,2,1),S,D,# ,(1,2,3),K_{SD}}$_{SK\text{-}D}$ (PK-D)

3→ 2 (Rp, SR(3,2,1),S,D,#,(1,2,3)) $(K_{SD})_{PK\text{-}S}$ { SR(3,2,1),S,D,# ,(1,2,3),K_{SD}}$_{SK\text{-}D}$ (PK-D)

2→1 (Rp, SR(3,2,1),S,D,#,(1,2,3)) $(K_{SD})_{PK\text{-}S}$ { SR(3,2,1),S,D,# ,(1,2,3),K_{SD}}$_{SK\text{-}D}$ (PK-D)

1→ S (Rp, SR(3,2,1),S,D,#,(1,2,3)) $(K_{SD})_{PK\text{-}S}$ { SR(3,2,1),S,D,# ,(1,2,3),K_{SD}}$_{SK\text{-}DS}$ (PK-D)

Figure 5. Message flow in the discovery of route {1,2,3} between source S and destination D with key exchange. *Legend*: ***** indicates broadcast address. SR(x,y,z) indicates source route x to y, y to z. SK-X and PK-X indicate secret key and public key of X respectively. $\{abc\}_{SK\text{-}X}$ indicates signature of X on abc. $(abc)_{PK\text{-}X}$ indicates encryption with public key of X. # indicates sequence number. Rq and Rp are request and reply tags. K_{SD} indicates secret key between S and D.

nor <random identifier, source S, destination D> triple as in SRP. Storing sequence number has the disadvantage that a malicious node can fabricate *Route Requests* with high sequence numbers and cause denial of service for future legitimate requests that now appear to be duplicate packets from that particular source. Although SRP prevents this false-replay attack by using a new random identifier for a source-destination pair instead of the sequence number, it cannot detect real-replay attacks since a malicious node can modify the random identifier and replay the packet as new one. Instead of maintaining state in intermediate route nodes, we maintain state only in end nodes; i.e., to detect replay of *Route Request* and *Route Reply* packets.

B. Symmetric-key version of our protocol

Given the limited computational and power constraints of some mobile devices, the use of public-key cryptography may impact the performance of the protocol both in computation and byte overhead to transport the public key. To limit this impact, we make minor modifications to the protocol aimed at minimizing the use of public-key cryptography.

We use public-key cryptography only when initiating a route discovery to a particular destination and only for the first time. During this interaction with the destination node, we exchange a symmetric key and use this symmetric key to maintain a route to this destination (i.e. for subsequent route discoveries caused by topology changes). Note that the destination node generates the symmetric key, encrypts it and binds the encryption cryptographically to its reply packet. The validity check performed by source S also ensures that the symmetric key is fresh; i.e., it is not an old key generated by D for this route. Figure 5 shows the flow of messages during route discovery using the modified protocol. Similarly, for route errors, if the node generating the route errors shares a symmetric-key with the source of the data packet it will use it to integrity protect the packet else it will use the its secret key. Therefore, we do not compromise the security of the secret-key associated with our SUCV identifier and also we can change the symmetric-key shared with a node, as often as needed.

C. Application to Other Protocols

Our solution for bootstrapping security associations between nodes for DSR can be used in conjunction with other secure routing protocols like SEAD [3] and Ariadne [5]. For example, SEAD assumes a shared key between all the nodes in the network to authenticate the source address of the updates sent by neighbors. Using the secure <IP address, public key> binding allows performing the same task without a pre-established shared key. Moreover, the last element of the one-way hash chain can be broadcast, and signed with the secret key associated with the public key of the secure <IP address, public key> binding. Similarly, Ariadne assumes that there exists a shared key between communicating nodes and that every node knows an element of the TESLA one-way key chain of every other

node. Using secure <IP address, public key> binding, communicating nodes can exchange a symmetric-key, and all nodes can broadcast the element of their TESLA one-way key chain signed with their secret key associated with the public key of the secure <IP address, public key> binding, without requiring any *a priori* secure association.

IV. SECURITY ANALYSIS

A. Security of <IP address, public key> binding

The security of the <IP address, public key> binding relies on the security of the public-key cryptosystem and on the second pre-image resistance property of the hash function. Both these assumptions are practical. For example, RSA public key cryptosystems would satisfy our security requirements (viz., Section II). Further, for a hash function that has an output of 64 bits and is second pre-image resistant (as both MD5 and SHA-1 are conjectured to be), an adversary must use 2^{62} attempts, on the average, to find a second public key that yields a given IP address (if we assume that one of the 64 bits is reserved, as in MIPv6). This work factor is clearly prohibitive for any practical adversary and any fast, second pre-image resistant hash function even if we ignore the fact that the adversary's attempts must be public keys and not arbitrary guesses of hash function inputs. Furthermore, although birthday attacks against the hash function may produce collisions between hash function inputs drawn from a uniform distribution after only 2^{32} attempts, such attacks are impractical in this setting. The number of nodes necessary to enable a collision between two arbitrary nodes would have to be of the order of 10^9 nodes, which is clearly impractical.

B. Security Analysis of the protocol

We consider several possible attacks mounted by non-colluding adversaries[2] on our protocol using the example topology of Figure 1. For simplicity we do not show the signatures and keys in the messages.

Attack 1: Assume that node 1 receives a route request from source node S. Node 1 may try to send a reply to S giving a false route. However, since node S expects a reply from destination node D, S will discard replies from any other node. Node 1 may try to pretend to be node D but the reply will not pass S's authentication check. Of course, node 1 can always drop the route request but that will be a problem when there is a single route from S to D and node 1 is on it.

Attack 2: Malicious nodes may try either to shorten or to lengthen a route by modifying the node list on a *Route Request*. For example, node 1 might not append its address to the list of a *Route Request*. Let us assume that nodes 2 and 3 follow the protocol correctly so that D receives the packet (Rq,S,D,#,(2,3)). D will construct a route reply and source route it over the reverse of the node list and when the route reply reaches node 2, it cannot send it to S as S is not its neighbor; so the reply will not reach S. Also, a malicious node might be able to lengthen a route by appending false IP addresses to a *Route Request* but it wouldn't gain anything other than the route being avoided, which it can achieve anyway by not forwarding the *Route Reques*t in the first place.

Attack 3: Intermediate nodes, such as node 2, might modify a *Route Reply* (e.g., it might add to or delete from the nodes of the node list), but S would not accept the modified *Route Reply* as a signature or the message authentication code of D would not pass S's authentication check.

Attack 4: An attacker might want to mount a replay attack. Replayed requests will be detected at D and replayed replies will be detected at S by using standard mechanisms based on sequence numbers.

Attack 5: An attacker can flood a node with route requests and exhaust its resources as the node has to authenticate packet signatures, and signature authentication is a computationally intensive operation. Alternatively, he can generate fake route errors to make a node verify the signatures. Most of these attacks are thwarted by the IP address check. To counter the rest of them, the protocol can be implemented in such a way that signature verification is the last check done. If any of the validity checks performed before signature authentication fail, the verifier can drop the packet.

V. COST ANALYSIS

We performed a set of preliminary experiments using the popular *ns-2* simulator with the *CMU Monarch* extensions for mobility to evaluate the overhead of our security mechanism. We used a random way point model for our mobility scenarios in the setting described in Table 1. We compare our protocol (SDSR) to DSR and an implementation of *DSR without optimizations* (DSR-NO-OPT).

[2] Like most other secure routing protocols[5,12], ours is not designed to handle attacks by multiple nodes acting in collusion, as would be necessary for the *wormhole* attack

Nodes	50
Scene	1000 m x 1000 m
Maximum velocity	20 m/s
Wireless range	250 m
Number of sources	10
Traffic	4 pkts/s

Table 1. Scenario for the *ns-2* experiments

A. *Packet Overhead*

SDSR does not support the DSR optimizations [7] since it performs end-to-end signature authentication of control messages and verification of whether a node is authorized to send a control message. Therefore, an intermediate node cannot reply from its cache or send a route maintenance message concerning a link that it is not an end-point of, since it cannot be verified whether that node had the right to send that message; i.e., the node could be malicious. In terms of packet overhead, there is no difference between SDSR and DSR-NO-OPT, since SDSR does not require any additional message exchanges. Figure 6 illustrates the packet overhead of DSR versus DSR-NO-OPT/SDSR.

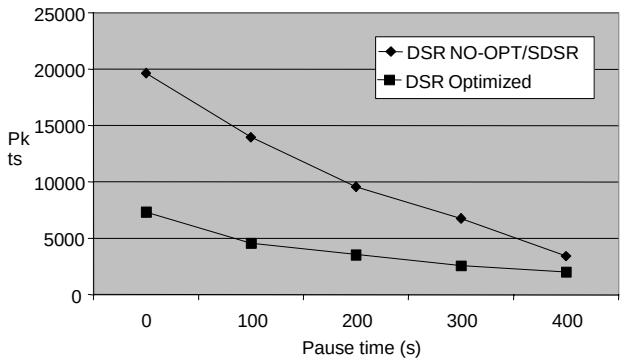

Figure 6. Packet overhead for DSR, DSR-NO-OPT/SDSR.

B. *Byte overhead*

SDSR exchanges the same number of messages as DSR-NO-OPT. However, these messages also contain signatures and public-keys. Every control message is signed (16 bytes) and contains the public-key of the signer (128 bytes) since it is not known whether the recipient already has a copy of the public key.

Alternatively, if the symmetric key version of SDSR is used and HMAC-MD5 is the message authentication code, then only an extra 16 byte field is used.

Figure 7 shows the byte overhead of DSR, DSR-NO-OPT, SDSR using only public keys, and SDSR using symmetric keys. This figure illustrates the significant overhead decrease in overhead when using the symmetric key version of SDSR instead of the public-key-only version.

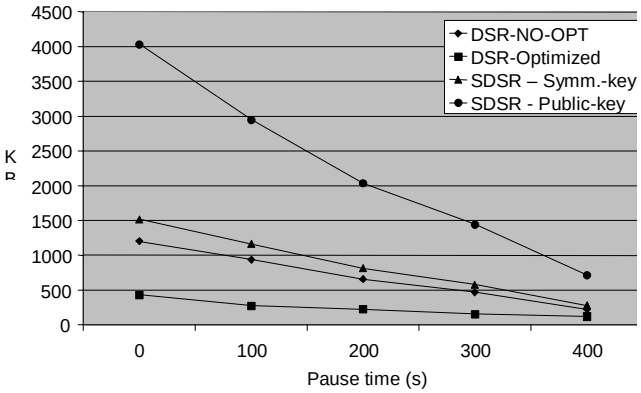

Figure 7. Byte overhead

C. *Average delay*

A lower bound. If the cost of security (hash, signatures generation and verification) were zero, our protocol would perform exactly as DSR-NO-OPT. This gives us a lower bound on the average delay of our protocol. By delay we mean the time between a packet is sent from the application layer at one node and received at the application layer of the destination node. This delay takes into account the route discovery and the transmission time. (We are still in the process incorporating the computation delay of the hash function and signature computation/verification into our *ns* simulations.) Figure 8 illustrates the average delay versus mobility. Note that, for high mobility, the non-optimized DSR performs slightly better than DSR. This

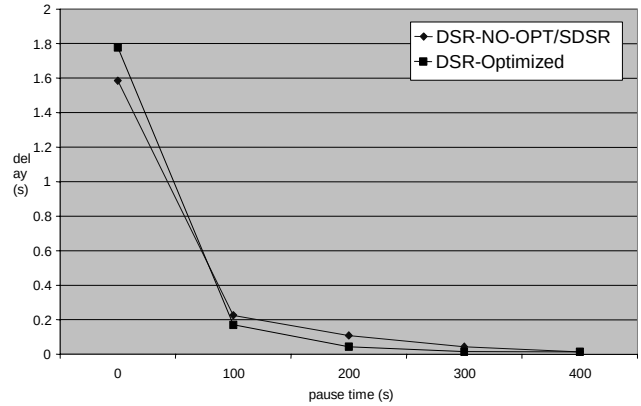

Figure 8. Average delay with respect to mobility

is caused by the high number of stale routes provided by the reply from cache in the optimized DSR [4]. However, for low mobility, DSR-NO-OPT has an average delay double that of DSR.

Computation-cost estimates. Prior work focused on the time and power consumption involved in signing and verifying a message using various algorithms [1,9]. For example, Modadugu et al. [9] show that on a 14Mhz Palm V the time to perform RSA operations is:

- 15 minutes, for 1024 bits RSA key generation;
- 27.8 seconds, for 1024 bits RSA signature generation, and
- 0.758 seconds, for 1024 bits RSA signature verification (e=3).

However, the PalmPilot was not designed with security in mind, and hardware crypto accelerators of the size of a dime, like the iButton [18], can now perform RSA operations in less than a second. (The cost of such a unit is about $15. Also, public-key generation can be done off-line.) Nevertheless, the computational cost can be decreased substantially by using symmetric keys since a message authentication code function such as the HMAC is very cheap to compute even on the Palm V.

VI. CONCLUSIONS

In this paper we proposed a scheme to bootstrap security within DSR thereby eliminating the need to assume pre-established secure associations among the nodes of the network. We achieved this through the use of a secure *<IP address, public-key>* binding. Our scheme is secure against multiple uncoordinated attackers. In future work we plan to evaluate our bootstrapping approach with other secure routing protocols in the presence of malicious nodes.

REFERENCES

[1] D. S. Wong, H. Ho Fuentes and A. H. Chan, "The Performance Measurement of Cryptographic Primitives on Palm Devices", Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001).

[2] Z. Haas, "A new routing protocol for the reconfigurable wireless networks", Proceedings of the IEEE International Conference on Universal Personal Communications, 1997.

[3] Y-C. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), IEEE, Calicoon, NY, June 2002 (to appear).

[4] Y-C. Hu and D. B. Johnson, "Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks", Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM, Boston, MA, August 2000.

[5] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks", Technical Report TR01-383, Department of Computer Science, Rice University, December 2001.

[6] J.-P. Hubaux, L. Buttyan and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks", Proceeding of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001), Long Beach, CA, USA, 2001.

[7] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks" in *Ad Hoc Networking*, edited by Charles E. Perkins, Chapter 5, pp. 139-172, Addison-Wesley, 2001.

[8] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks", Proceedings of the Sixth annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 255--265, 2000.

[9] N. Modadugu, D. Boneh, and M. Kim, "Generating RSA keys on a handheld using an untrusted server", in RSA 2000, 2000.

[10] G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses", Proceedings of the 2002 Network and Distributed System Security conference (NDSS02), San Diego, February 2002.

[11] G. Montenegro and P. Nikander, "Protecting Against Bidding Down Attacks". Internet Draft, draft-montenegro-mipv6sec-bit-method-00.txt, April 2002. Work in Progress.

[12] G. O'Shea and M. Roe, "Child-proof Authentication for MIPv6 (CAM)", ACM Computer Communication Review, April 2001

[13] P. Papadimitratos and Z. Haas, "Secure Routing for Mobile Ad-Hoc Networks", Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS2002), San Diego, CA, January 2002.

[14] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks", in IEEE Infocom, 1997, pp. 1405-1413.

[15] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", Proceedings of the ACM SIGCOMM, October 1994.

[16] C. E. Perkins and E. M. Royer, "Ad hoc On-Demand Distance Vector Routing", Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90-100.

[17] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science, 1999.

[18] iButtonHomePage, http://www.ibutton.com.