MASTER'S THESIS

Source Authentication for Multicast in Mobile Ad hoc Networks

by: Prabha Ramachandran
Advisor: John S. Baras

MS 2003-6

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2003		**00-00-2003 to 00-00-2003**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Source Authentication for Multicast in Mobile Ad Hoc Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland,The Graduate School,2123 Lee Building,College Park,MD,20742**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**107**	

ABSTRACT

Title of Thesis:	SOURCE AUTHENTICATION FOR MULTICAST IN MOBILE AD HOC NETWORKS
Degree candidate:	Prabha Ramachandran
Degree and year:	Master of Science, 2003
Thesis directed by:	Professor John S. Baras Department of Electrical Engineering

Recent emergence and popularity of mobile ad hoc networks in a host of current-day applications has instigated a suite of research challenges, primarily in routing and security issues for such networks. The ease and low cost of deployment make this networking paradigm very convenient for group-oriented applications like battlefield missions, business conference, virtual classroom, etc. Such networks are characterized by wireless "links", lack of any fixed network infrastructure, rapidly changing topology and mobile hosts.

Security for these dynamic ad hoc networks presents many challenges in the area of multicasting for group-oriented tactical missions, operating in a hostile environment. Key-management and secure-routing have been the primary

research focus in this area. Source authentication for multicast is also a fundamental problem that needs to be addressed.

In this work, we study some of the proposed source authentication schemes for multicast group communication and evaluate one such scheme for a tactical ad hoc set-up. We propose solutions that exploit the hierarchical nature of tactical networks to achieve time synchronization pre-requisites that the proposed schemes have.

We define metrics to evaluate the authentication scheme and present simulation results for the authentication scheme evaluated with two different time synchronization techniques. We find that our selected authentication scheme is well suited for a mobile ad hoc network. We show that our solution for time synchronization performs much better than conventional methods suggested for the authentication scheme. We also discuss applications of our overlay architecture for bootstrapping the authentication scheme with reduced communication overhead.

SOURCE AUTHENTICATION FOR MULTICAST IN MOBILE AD HOC NETWORKS

by

Prabha Ramachandran

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2003

Advisory Committee:

Professor John S. Baras, Chair
Professor Virgil Gligor
Assistant Professor Sennur Ulukus

DEDICATION

To my parents and sister.

ACKNOWLEDGMENTS

The work presented in this thesis was initially suggested and continually supported by my advisor, Dr. John Baras. I am grateful for his guidance, support and encouragement. I would like to thank Dr. Virgil Gligor and Dr. Sennur Ulukus for agreeing to serve on my committee and for reviewing my thesis.

Sincere thanks to Huigang Chen for helping me with the simulations in the initial stages, and to Karthik Chandrasekar, Dinesh Dharmaraju and Manish Karir for helping me fix bugs in the MAODV code. I owe many thanks to my colleagues Ayan Roy-Chowdhury and Tao Jiang for critiquing and helping me refine many of the ideas presented in this thesis. I would also like to thank Harsh Mehta and Nalini Bharatula for their suggestions and the rest of the SEIL lab and CSHCN for creating a conducive work environment. I am grateful to my parents, sister and Madhavi for their constant support and encouragement.

I gratefully acknowledge the financial support for my Masters through a Graduate Research Assistantship from the Institute for Systems Research. The work reported in this thesis was prepared through collaborative participation in the Collaborative Technology Alliance for Communications & Networks

sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0011.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

A mobile adhoc network (MANET) [1, 2, 3, 4, 5, 6] is a dynamically changing multi-hop network created by a set of wireless mobile nodes that cooperatively and spontaneously interact without any fixed infrastructure or centralized authority. A node communicates directly with nodes within wireless transmission range and indirectly with all other destinations using a dynamically-determined multi-hop route, relying on other nodes to act as routers. These networks can be formed, merged and partitioned on the fly.

MANETs can be of different kinds depending on the application. It is possible to use very small and common devices as nodes. For example, an ad hoc network can be formed in a conference room between PDAs, laptops and cellular phones of the members. A completely different scenario could be a group of soldiers, planes and tanks operating in a hostile environment. Tactical military operations are by far the most widespread application of MANETs. Figure 1.1 shows a typical tactical MANET. These networks are hybrid and hierarchical in nature. Still, much of the research in the area of ad hoc networks assumes a peer

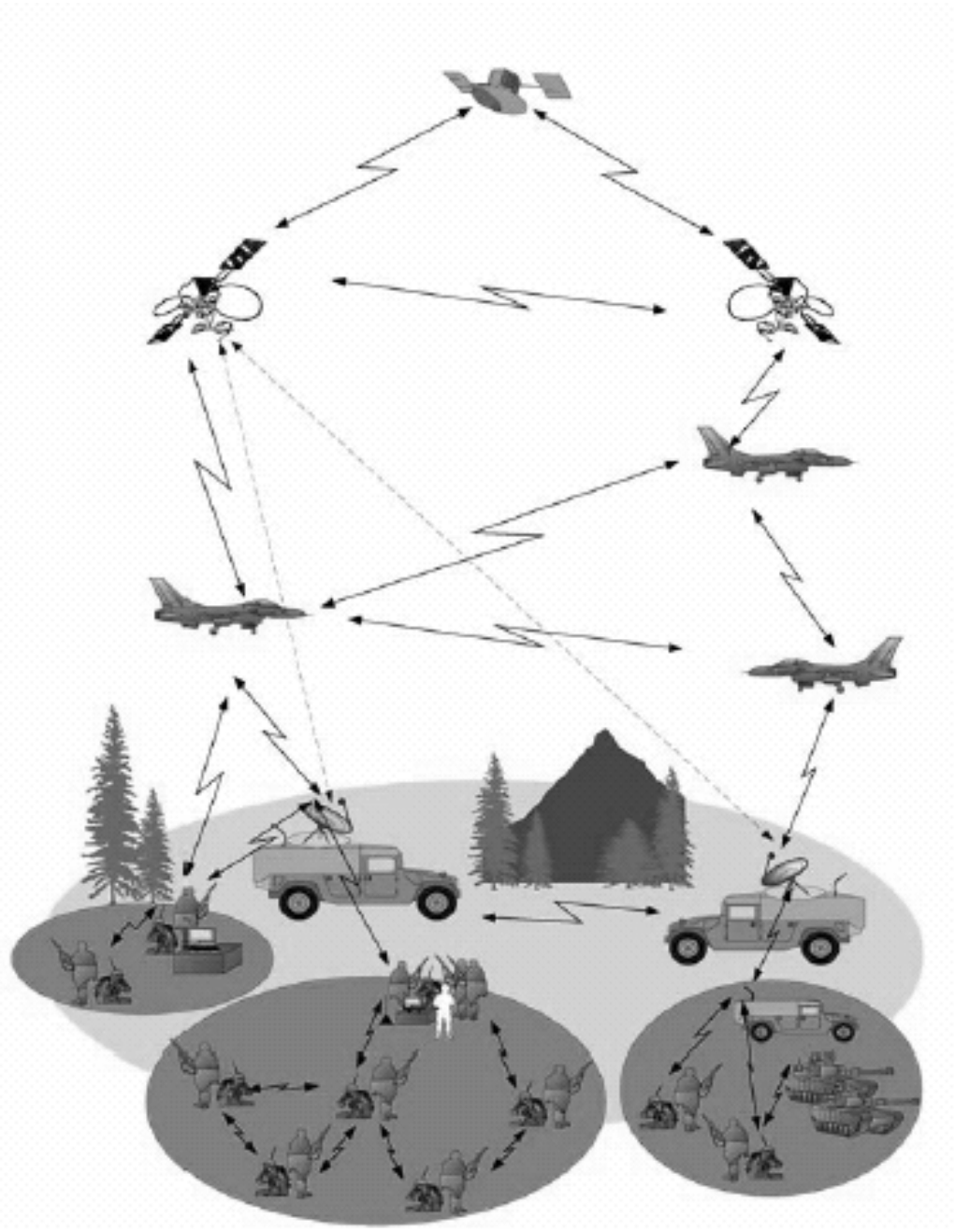


Figure 1.1: A tactical MANET: Soldiers, planes, copters and tanks operate in a battlefield in a hierarchical fashion, forming groups at different levels

to peer setting.

Other potential applications of MANETs include disaster response, audio and video conferencing, gaming applications and various forms of ubiquitous computing.

The following are the salient features of a MANET –

- Dynamically changing topology
- Poor physical protection of nodes
- Bandwidth limitations
- Energy constrained nodes
- Wireless communication medium and mobile hosts

1.1 Ad hoc network security

Topology changes, bandwidth limitations and energy constraints pose various problems in the absence of a fixed infrastructure in hybrid, dynamic ad hoc wireless networks. Securing such MANETs is a particularly challenging task because of the vulnerability of the wireless "links", poor physical protection of the nodes, the sporadic nature of connectivity and the dynamically changing topology. The absence of a certifying authority and the lack of a fixed infrastructure or a centralized management point make the task all the more challenging [4].

1.1.1 Threats and vulnerabilities

The threats and vulnerabilities of MANETs have been discussed in detail in [1, 3, 4]. We summarize below the main challenges.

- Attacks due to vulnerability of wireless "links" –

  These include passive eavesdropping, active impersonation, message replay, message distortion, injection of erroneous messages, congestion or denial of service, etc.

- Attacks due to compromised nodes –

  Poor physical protection of nodes in a hostile environment results in a non-negligible probability of a node being compromised. Attacks launched from within the network by compromised nodes should be taken into consideration. The security mechanisms should not be based on a central entity to avoid single point(s) of failure in the network. It is also important to ensure that the system fails gracefully. In general, the failure of a single node should not jeopardize the security of the system as a whole.

- Attacks due to vulnerability of security mechanisms –

  These include malicious replacement of public keys, compromise of keys, etc. The security mechanisms should be resistant to both passive and active adversaries. Recently, there have been a number of side-channel attacks on many popular cryptosystems. These attacks do not break the

security mechanism as such, but use information like padding in messages, key-strokes, power analysis, etc to recover the key. The key recovery is not due to the weaknesses in the security mechanism. Examples include electromagnetic and differential power analysis techniques to break smart-card security and padding attacks on Cipher-block-chaining mode of encryption.

- Denial of Service threats –

  A Denial of Service (*DoS*) can be created by malicious nodes or by unintentional failure of node(s) in the network. Forms of *DoS* vary from a simple flooding of a resource to congesting parts of the network to reconfiguration of routing protocol by malicious nodes leading to Byzantine failures.

1.1.2 Security-related Requirements

The security requirements depend largely on the mission and operating conditions of the MANET. For military applications, the requirements will be stringent in terms of confidentiality and resistance to DoS attacks. For example, we might have to ensure that wireless traffic does not reveal the location of a target to the enemy in a military mobile ad hoc network while it is not necessary to do so in a civilian ad hoc network. The following are the general security requirements in a MANET [1, 3, 4]. Depending on the application, the

constraints and requirements would eliminate or make more critical one or more of the following or have added restrictions.

- Authentication –

  The receiver of a message must be able to verify the identity of the originator of the message, thereby preventing an intruder from masquerading as a legitimate source of the message.

- Integrity –

  The receiver of a message should be able to check whether a message was modified in transit. This is to avoid accepting packets that have been modified by a hostile node, while in transit.

- Confidentiality –

  Confidentiality is needed to ensure that certain information (like tactical military information or routing information that might reveal the location of the target) is never disclosed to unauthorized entities. Confidentiality is required only when the message is to be kept secret.

- Non-repudiation –

  The originator of the message cannot deny having sent the message.

  Non-repudiation is useful for detection and isolation of compromised nodes.

- Availability –

  Availability ensures functionality of network services despite *DoS* attacks.

The security mechanisms should not make any assumptions about the availability of specific nodes at any given time. Nodes may be idle, shut down for a while or could be compromised.

- Access Control –

  Some systems may require that only certain nodes be entitled to perform certain tasks, e.g posting certain messages (commands/orders), revoking a certain policy / trust information, etc. Access control mechanisms must be established to deal with these issues.

- Scalability –

  Security mechanisms may be required to be scalable to handle a large number of nodes, depending on the application.

- Adaptability to changes –

  Frequent changes in topology and membership make the ad hoc network dynamic. Trust relations among nodes also change with time and so does a node's affiliation with an administrative domain. The security mechanisms should not have a static configuration and should be able to adapt to changes on the fly.

- Key Management –

  The only way to accurately enforce authentication, integrity and non-repudiation is by using some form of cryptography, which requires the

distribution/exchange of encryption/authentication key information among message senders and receivers. A key management module consists of a trust model, a cryptosystem, key establishment module, a key storage and key distribution service. Some group-oriented applications may require key revocation services as well.

1.2 Multicast in MANETs

Multicast means being able to deliver a packet to a set of destinations. Multicast communication enables one-to-many or many-to-many communication as opposed one-to-one supported by unicast. A source can send the same packet simultaneously to multiple destinations without any repetitive transmissions. In IPv4, a multicast address is designed to enable the delivery of datagrams to a set of hosts that have been configured as members of a multicast group across various subnetworks [7, 8].

Multicasting allows a sender to transmit voice, video and text to multiple receivers simultaneously. Broadcast is one-to-all communication and is a special case of multicast. When sending the same packet to multiple receivers, multicast improves bandwidth utilization and involves lesser host/router processing. The typical applications of multicast are multi-party video or audio conferencing, resource discovery, news feeds, online games, TV/video transmissions (Pay Per View TV), etc. In most ad hoc networks, nodes work in groups to perform a

given task and hence multicast plays an important role. Several multicast routing algorithms have been proposed for MANETs. These include MAODV[9], ODMRP[10], MCEDAR[11], AMRoute[12], MOSPF[13] etc.

1.2.1 Multicast security

Multicast security issues and proposed solutions have been studied in [14, 15]. The primary objectives of a multicast security infrastructure are to maintain secrecy and guarantee authentication for all group communication so that only legitimate senders can multicast packets to the group and only packets sent by legitimate group members are accepted. Other security concerns include anonymity, non-repudiation, access control, trust issues, maintaining service availability to protect the network from clogging attacks, etc. Security in multicast is thus considerably more complicated than in the unicast case. Most unicast solutions are prohibitively inefficient for multicast scenarios.

Factors affecting security [15] are group type, group size, member (node) characteristics (power, storage, availability), membership dynamics, membership control, number and type of senders, volume and type of traffic and routing algorithm used. Attacks on routing mechanisms are becoming widespread. Thus multicast security is a fairly complex multi-faceted, multi-layered problem. These requirements are even more difficult to fulfil in ad hoc networks where bandwidth, storage and energy constraints of the nodes pose additional problems

when coupled with mobility and dynamically changing topology in the absence of a centralized infrastructure.

1.3 Motivation and Problem Definition

In group communications, establishment of a shared session key does not "secure" communication within the group. If messages are encrypted with the session key, then confidentiality (privacy) of the message is guaranteed. This alone does not prevent a legitimate group member from masquerading as another member. There are several routing-related attacks that any node (group member in the context of multicast) can cause in the absence of authentication. These have been discussed in [16]. These include, but are not limited to, injecting erroneous routing messages, selectively forwarding only routing packets, creating black holes, worm-holes , creating *DoS* attacks, causing a vertex cut or network partition, etc. These attacks are however not specific to multicast settings. Even if the multicast routing protocol is secure against all possible routing-related attacks, there is nothing to prevent one group member from spoofing as another. Hence, multicast security is not guaranteed by "securing" the routing and having a "secure" group key management scheme. Source authentication is an important security concern that needs to be addressed.

Authentication in a group can be of two types [14] – Group Authentication and Source Authentication. Group authentication means that a group member

can distinguish the sender of the message as a group member or non-member. Source authenticity means that a member can identify the purported sender of the message within the group (or outside).

Our goal is to study existing source authentication schemes and explore if any of the proposed schemes can be used to guarantee source authenticity for multicast in a tactical MANET, modify or propose new schemes if necessary and to quantify the candidate scheme by simulation in a realistic multicast setting in wireless, mobile, ad hoc mode.

For example, in a tactical MANET, an enemy aircraft can spoof its identity and try to join a group. If it is in the vicinity, it can eavesdrop on all communication in range. If a node is compromised by the enemy, it can learn the session key and the enemy can silently listen to and intercept all communication and pose as a legitimate soldier/node. Worse still, it can give false "orders" (e.g to dissolve the group and retreat, to lie in ambush in a different place, etc). All these attacks can be prevented with source authentication. The problem of detecting compromised nodes without any central authority is hard, especially in a MANET and is beyond the scope of this work. We assert that our goal is to propose a candidate algorithm for source authentication in multicast and evaluate its performance.

1.4 Contributions

We propose a source authentication scheme, Tesla, suggested by Perrig et al [17, 18] as a candidate for achieving source authentication for multicast in a MANET. We define metrics to evaluate the scheme and measure the performance by simulations in different scenarios for a MANET. Our results show that the scheme is suitable for group communication in MANETs. We describe means to achieve some strong assumptions that Tesla is based on. We use a combination of μTesla [19] and extensions to the basic Tesla scheme [17, 18] with additional support for tactical mobile ad hoc networks. We also define a framework which helps achieve some pre-requisites for the authentication scheme and also helps in bootstrapping the authentication process. We simulate the authentication scheme in Network Simulator [20], version 2.1b9a.

1.5 Organization of Thesis

The rest of the thesis is organized as follows. In Chapter 2, some of the existing multicast authentication schemes have been discussed. We propose one of these schemes as a candidate source authentication scheme for multicast in MANETs. However, this scheme requires loose clock synchronization among participating nodes. Chapter 3 discusses the problem of time synchronization in a MANET. We give details of the architecture proposed to efficiently bootstrap and synchronize nodes in Chapter 4. Chapter 5 describes the simulation framework

and results. We present a brief qualitative security analysis along with the conclusions in Chapter 6.

Chapter 2

Related Work

As discussed in Section 1.3, source authentication involves authenticating the purported sender (not necessarily a group member) of the message. Criteria for judging source authentication solutions have been discussed in [14, 15]. These include efficiency, reliability requirements, collusion-resistance, latency or delay in authentication. In addition, non-repudiation, revocation, updating should be feasible with minimal communication and computation overhead. Entity authentication should provide for evaluation of the authenticity of an entity and computation of trust values (opinions) conforming to the trust metrics.

Our goal is to develop a scheme that is a combination or improvement of one or more of the existing schemes with additional support and modifications needed for the ad hoc tactical set-up. In this work, we concentrate on schemes for source authentication for group communication. We are not trying to secure routing. Our scenario is a dynamic multicast group in a MANET, where members join and leave on the fly. We assume the existence of a valid session key (for data confidentiality) at any instant of time (it is updated as and when any

change in membership occurs in the group). Source authentication is still needed for the messages sent within the group and for messages sent to the group by a non-member. We assume message authentication in this case to be equivalent source authentication. Each receiver needs to know the "correct" identity of the sender.

In the rest of the chapter, we briefly outline previous solutions for the problem and discuss two recent schemes, Tesla and μTesla in detail.

2.1 Previous Solutions

Authentication may be provided in many forms with varying degrees of assurance from simple passwords transmitted in the clear to digital signatures based on public key cryptography. For the unicast case, authenticity and integrity can be guaranteed by using keyed *message authentication codes* (MACs).

- Sender and receiver share a secret key K.
- Sender sends $< data, MAC_K(data) >$
- Receiver gets $< data, c >$ and verifies if $c = MAC_K(data)$

In case of multiple receivers, the problem becomes much harder to solve because a symmetric approach would allow anyone holding a key (that is, any receiver) to forge packets. Several solutions have been proposed for the multicast case

[21, 22, 23, 24]. These have been reviewed in [14, 15, 25]. We briefly summarize the proposed solutions.

- Sign each packet using public key signatures – Secure, but expensive in terms of key generation and signature verification overhead.

- Stream signatures [23] – Only one regular signature is transmitted at the beginning of the stream. Each packet contains a cryptographic hash of the next packet. This scheme cannot tolerate packet loss. If the packet stream being sent is not known apriori, the sender has to embed one-time keys and signatures into the packet stream, resulting in a large overhead.

- Tree-based signatures [22, 24] and hybrid signature schemes [21] – Resistant to collusion and packet loss, but size overhead is very large.

- Multiple MACs [15] – Sender maintains several keys, subsets of which are shared amongst receivers in such a way that no subset of N receivers knows all the keys known to any other receiver. This scheme is insecure when there are more than N colluding parties. While N should be fairly large to prevent collusion, a large number of MACs need to be concatenated to the message, increasing the overhead on each packet.

The principal issues for a source authentication mechanism for multicast [17, 18, 26] include loss tolerance, high efficiency, and no per-receiver state at the sender. The problem is particularly hard in settings with high packet loss rates

and where specific packets in the sequence that are lost are not retransmitted, and where the receiver wants to authenticate every packet that it receives . With many receivers, we typically have a high variance in the bandwidth of the receivers, with high packet loss for the receivers with relatively low bandwidth. Nevertheless, we want to assure source authentication to work even in the presence of wide variation in bandwidth among nodes. In the following sections, we describe two such schemes proposed by Perrig, et al [17, 18, 26], which satisfy several of the above requirements.

2.2 Tesla

Tesla (Timed Efficient Stream Loss-tolerant Authentication) [17, 18, 26], uses only symmetric cryptographic primitives such as pseudo-random functions (PRFs) and message authentication codes (MACs), and is based on delayed disclosure of keys by the sender. The sender and the receiver are required to be loosely time synchronized. Bootstrapping is done through a regular data authentication system. A digital signature algorithm is used for this purpose, in which case the receiver is required to have an authentic copy of either the sender's public key certificate or a root key certificate in case of a PKI (public-key infrastructure). The MAC and the PRF must be cryptographically secure.

The basic scheme works as follows. The sender issues a signed commitment to a key K_i that is known only to itself. The sender then uses that key to compute

a Message Authentication code (MAC) on a packet P_i and later discloses the key K_i in packet P_{i+1}, which enables the receiver to verify the commitment and the MAC of packet P_i. If both verifications are successful, packet P_i is authenticated. The commitment is realized via a pseudo-random function, F with collision resistance. The MAC key is derived from the key revealed in the packet using another PRF F'. $K'_i = F'(K_i)$ is the secret key used to compute the MAC of the next packet, and $F(K_i)$ commits to the key K_i without revealing it. The functions F and F' are two different pseudo-random functions. Commitment value $F(K_i)$ is important for the authentication of the subsequent packet P_i. To bootstrap this scheme, the first packet needs to be authenticated with a regular digital signature scheme, for example RSA [27]. When receiver R receives P_{i+1}, K_i is disclosed and R first verifies that K_i is correct by checking if $F(K_i)$ and the commitment sent in packet P_i match. R computes $K'_i = F'(K_i)$ to check the integrity of P_i by verifying the MAC in P_i. Similarly, P_{i+1} is authenticated after the receipt of P_{i+2} and so on.

There are a number of problems with this scheme. First it is not resistant to packet loss. If a subsequent packet is lost, then the previous packet cannot be authenticated. Robustness to packet loss is achieved by using a one-way key chain instead of using an independent key for each packet. The sender first chooses a random key K_N and computes $K_i = F(K_{i+1})$ for $i = N-1, \cdots, 1$, where N is the length of the key chain. $K_i = F^{N-i}(K_N)$ for $i = 1 \cdots N-1$. It uses the keys in reverse order, i.e. the key, K_0 is used to authenticate the first packet so that

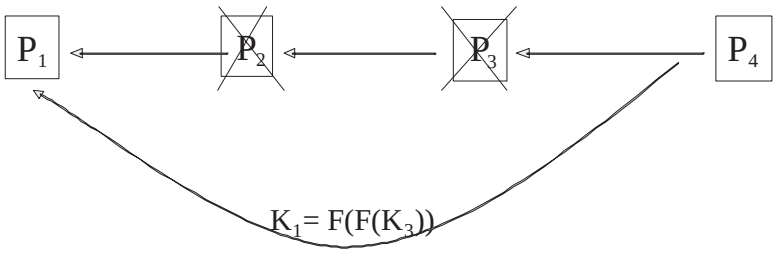


Figure 2.1: Achieving robustness to packet loss: Use of a one-way key-chain enables the receiver to authenticate packet P_1, after receipt of P_4 when P_2 & P_3 are lost.

when the next key, K_1 is revealed, the key-authenticity can be verified by checking if the hash (PRF) of the key revealed equals the key revealed in the previous packet. Figure 2.1 shows an example to illustrate this concept. All previous keys can be computed from a given key by repeated application of the PRF F. This means that any key can be used to authenticate subsequent keys, i.e. the receiver(s) can efficiently authenticate K_j given $K_i (i < j)$ by verifying if $K_i = F^{j-i}(K_j)$. It is infeasible to derive K_j from K_i for $(j > i)$. These properties of the one-way key chain make the scheme robust to missing values (packets).

Even now, the scheme is not secure. What if the receiver gets packet P_{i+1} before packet P_i? What if a man-in-the-middle attack is launched by an adversary? The adversary can hold on to P_i until it gets P_{i+1}. The key K_i is revealed in P_{i+1} and the adversary can modify the message and compute the MAC using the key and send these two packets one by one to the receiver. Thus it is clear that receiver should be able to verify that the key used to compute the MAC of a packet it received has not been disclosed by the sender until the time

of receipt. This is called the "security condition". This is possible if the sender and receiver are loosely clock synchronized and the receiver knows the sender's key disclosure schedule - i.e the time of disclosure for a particular key.

If each packet discloses the key corresponding to the MAC of the previous packet, then all packets received after a delay greater than the sender's key-disclosure-delay will be dropped by the receiver(s) when they try to verify the key-authenticity as described above. To make the scheme adaptive to network delays, the key is disclosed after d packets. The key K_i used to compute the MAC of packet P_i is disclosed in packet P_{i+d}.

The receiver(s) however need to know the exact sending schedule of the sender since the key-usage and disclosure is on a per-packet basis. To accommodate dynamic sending rates, a key is tied to a period in time (one time interval) and the same key is used to compute the MAC for all packets sent in a given interval. The key K_i is used to authenticate packets sent in interval i and is disclosed in interval $i + d$. Figure 2.2 illustrates the concept of using one key per time interval. The length of the key chain is significantly reduced by using one key per interval.

The choice of the parameter d is critical. A small value will make remote receivers drop the packet. A large value of d increases the time to authenticate a packet, requiring large buffer storage at the receiver. Clearly, there is a trade-off. Simultaneous use of multiple authentication chains with different disclosure periods helps accommodate heterogeneous receivers across the network. This enables each receiver to use the chain with minimal disclosure delay, sufficient to

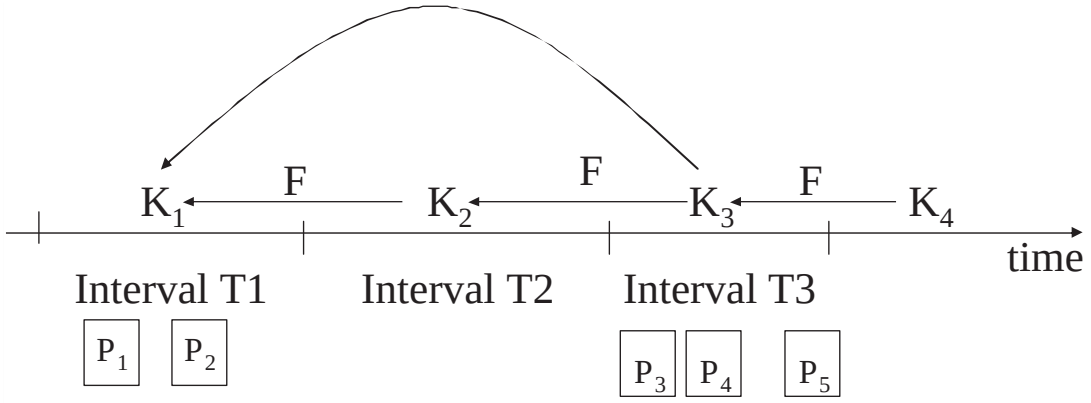


Figure 2.2: Tolerating dynamic sending rates: The figure shows the one way key-chain and packets being sent at the sender. In this example, K_1' derived from K_1 is used to authenticate packets P_1 and P_2 and is disclosed in packets P_3, P_4, P_5 sent in interval 3.

prevent spurious drops, which are caused if the security condition does not hold. When operating in an environment with heterogeneous network delays for different receivers, Tesla authenticates each packet using multiple keys, where the different keys have different disclosure delays. This results in larger overhead, both in processing time and in bandwidth. Using the same key chain for all instances with a different key disclosure schedule for each instance reduces the communication overhead. These extensions are discussed in detail in [17, 18, 26].

To summarize, Tesla has the following properties:

- Low computation overhead – The authentication involves typically only one MAC function and one hash function computation per packet, for both

sender and receiver.

- Low per-packet communication overhead – Overhead can be as low as 10-20 bytes per packet.
- Resistant to packet loss – Every packet that is received in time can be authenticated.
- No sender-side buffering – every packet is sent as soon as it is ready.
- High guarantee of authenticity – The system provides strong authenticity. By strong authenticity, we mean that the receiver has a high assurance of authenticity, as long as the timing and cryptographic assumptions are enforced.

Tesla is tailored for multicast. A new group member only needs to synchronize its time with the sender and receive the sender's key disclosure schedule along with a commitment to the key chain. An initial authenticated packet is still required to bootstrap the authentication process. Tesla uses a digital signature based periodic broadcast scheme for this purpose. The sender periodically broadcasts an interval specification message, digitally signed with its private key. The initial authenticated packet contains the precise interval information (interval frequency and the starting time of a specific interval) along with the key of a past interval.

2.3 μTesla

μTesla [19] was used in sensor networks for authenticated broadcast. μTesla uses only symmetric mechanisms to authenticate the initial packet unlike Tesla, which uses digital signatures, which are too expensive for resource-constrained environments. μTesla uses the node-to-Base station authenticated channel to bootstrap the authenticated broadcast. The sender uses one key per epoch and discloses the key in a separate key disclosure packet. A single key is used to compute the MAC of all packets sent in a given interval. μTesla restricts the number of authenticated senders. Loose time synchronization between the nodes is assumed.

Two modes of communication are supported by μTesla.

1. Base station (BS) broadcasts authenticated packets to nodes (BS is the Sender)

    - Sender Set-up: BS chooses a random key and generates a one-way key chain.
    - Broadcasting authenticated packets: BS associates each key of the key-chain with one time interval t. BS uses the current key K_t to MAC packets sent in interval t. BS reveals the key K_t after a delay of "key-disclosure-delay" intervals after the end of time interval t.
    - Bootstrapping a new receiver: Receiver sends a nonce with its request

to the sender (BS) who replies with its current time, a key K_i of the key-chain, the key disclosure schedule (starting time of interval i, duration of a time interval, the disclosure delay) along with a MAC using the shared key between the node and the BS.

- Authenticating broadcasts: Receiver first verifies the security condition when it receives a packet. It then authenticates the key and finally verifies the MAC when it receives the disclosure packet revealing the MAC key.

2. Nodes broadcast authenticated data (Any node is the sender)

   - A node broadcasts data through the base station using the secure channel between the node and the BS to send data in a reliable way to the BS, which then broadcasts it.
   - Alternatively, a node broadcasts the data while the BS keeps the one-way key-chain and sends keys to the node as and when needed. To conserve energy at the broadcasting node, the BS can also broadcast the key disclosure packets and perform the initial bootstrapping.

The two schemes described in this chapter provide efficient means to authenticate the source. They differ in their bootstrap mechanisms and key-usage. However, loose clock synchronization among the participating nodes is a pre-requisite for both. This is a very strong assumption. Some general

solutions are outlined in [17, 19, 26]. Some of these solutions are not applicable to a MANET. In the following chapter, we discuss time synchronization issues in a MANET. We discuss the pros and cons of some of the proposed solutions and suggest some for our tactical set-up.

Chapter 3

Time Synchronization in a MANET

In Tesla and other applications like SPINS [19] and Ariadne[16], loose clock synchronization between nodes is assumed at the start. *Loose time synchronization* means that the synchronization does not need to be precise, but that the receiver(s) must know an upper bound on the dispersion (the maximum clock offset) between the sender's clock and itself. The solutions outlined in [17, 18, 16, 26] are not suitable for a MANET. In a tactical MANET, the requirements and constraints are different.

In this chapter, we describe the requirements for achieving loose clock synchronization in a MANET in order to implement Tesla. Next, we discuss some of the common time synchronization schemes and reason why each of those would not fare well in our case. We also mention some possible attacks that should be taken into consideration for a tactical MANET. We then proceed to exploit the hierarchical nature of present-day tactical networks and emphasize that an overlay will help not only in time synchronization, but also in routing and key distribution.

3.1 Requirements and Related Work

All nodes in the MANET should be loosely clock synchronized, i.e. each node should know a rough upper bound on the dispersion between its clock and that of a sender from whom it receives data in the group. The receiver need not know the real difference δ, between the sender and receiver's time but only some $\triangle$ that is guaranteed to be greater than or equal to δ [17, 18, 26]. Ideally, any time synchronization scheme that is robust against active adversaries can be used. Tesla supports two synchronization methods, direct and indirect synchronization. In direct synchronization, the receiver synchronizes its time directly with the data sender while in indirect synchronization both the sender and the receiver synchronize their time with a common (external) time synchronization service.

In the basic protocol described in [17], the receiver sends a request to the sender with a nonce (to prevent replay). The sender replies with its current time t_S along with its key disclosure schedule and an initial commitment to its key-chain. This packet is signed with a regular signature scheme. The maximum time discrepancy, d_t is one Round Trip Time (RTT). This solution however does not scale well. There is a bottleneck at the sender when there are a large number of receivers.

Direct time synchronization has a few disadvantages over indirect time synchronization in terms of communication overhead and the possibility of a Denial of Service (DoS) attack at the sender. This is clearly avoided in indirect

synchronization where the nodes synchronize their time with a common external trusted time synchronization service.

An alternate solution using secure distributed time-servers is described in [18]. Initially, the sender synchronizes its time with the time-server and computes the maximum synchronization error. It then broadcasts the maximum synchronization error periodically along with the interval information. Receivers can independently synchronize their time with the synchronization server and individually compute their maximum synchronization error. The receivers add up all the synchronization error values to verify the security condition. A third solution is proposed in [26] for the multicast scenario. Senders and receivers synchronize their time with time-servers dispersed across the network after which every node knows the time and the maximum synchronization error. The sender periodically broadcasts a signed bootstrap packet along with its offset to the server's clock. A receiver waits for a bootstrap packet. It computes the synchronization error between itself and the sender as the sum of the synchronization errors of the sender and itself. This works if the sender and receiver have a method to synchronize time and the receiver knows the upper bound of the synchronization error. The time-servers distributed across the network should be synchronized with each other.

The mobility and adhoc-ness involved in MANETs make each of the above methods inefficient and practically un-scalable. This is tantamount to saying that the difference in clocks between any two nodes should not be more than a

threshold. Each node should know a rough upper bound of the maximum clock difference that can be tolerated by the authentication scheme (or any other application for that matter). Loss of a time synchronization packet should not prevent other nodes from updating their timing information. This implies that we cannot use a concept of "global" time, where nodes update their time based on the clocks of their peers. This is not practical for MANETs.

For each multicast group, an associated group leader maintains the tree in case of MAODV and other tree-based protocols. The group leader could send out beacons containing timing information periodically. The receiver need not contact the sender in this case. It just waits for a beacon from the group leader. This method is not resistant to packet loss. A major discrepancy arises if a node is a member of two groups at the same time whose senders have a huge clock difference. The group leader itself is likely to change with dynamic changes in topology. When a non-member sends a packet to the group, this method fails and one needs to revert to the aforementioned methods.

Many time synchronization services for sensor networks use Reference Broadcast Service (RBS) [28] as a means to achieve time synchronization. All these methods involve message exchanges between the receivers. This is not efficient and scalable for a MANET. [29] discusses time synchronization techniques for ad hoc networks.

Taking all the issues discussed in this section into consideration, we arrive at the following set of requirements:

- All nodes should be loosely time synchronized with each other, since any node can be a sender.
- No message exchanges – A node should not have to send a message to the time-server or sender for synchronization.
- A node should not depend on "many" other nodes for updating its clock.
- Timing information should be refreshed periodically if necessary.
- The time synchronization packet should be reliable - needs source authentication and integrity guarantee. Confidentiality is not required.
- Congestion and collision should not pose major problems.

3.2 Threats

We also need to keep in mind the possible attacks on such a time synchronization service. [30] examines the security requirements of accurate time services and analyses the security of NTP [31]. The following are the common attacks on a time-synchronization service, as discussed in [30].

- *Masquerade* – An attacker can try to impersonate the time-server.
- *Tamper* – Attacker can modify the packet containing timing information.
- *Replay* – Attacker can replay (re-send) old messages.

- *Denial of Service (DoS)* – The attacker can intercept the time-server's messages and delete them. The attacker can flood the network with spurious packets or intercepted packets.

- *Delay* – The attacker can intercept and hold on to these packets and release them after sometime, introducing long and often unacceptable transmission delays.

3.3 Solutions

Current day solutions for time synchronization in computer networks [32] include: Terrestrial communication systems like TV and telephone (modems), Direct radio broadcasts (WWV, WWVH), Navigation systems like GPS, Loran-C, Satellite Communication Systems like Two-Way Satellite Time Transfer (TWSTT), etc. We briefly discuss some of these with their pros and cons.

3.3.1 Satellite

Suppose we have a satellite that periodically broadcasts the timing information to all the nodes of the MANET. In a typical battlefield scenario, a MANET would range over 25Km X 25km. A single satellite can cover this area and can broadcast timing data in a bandwidth-effective and cost-effective manner. Satellites have been effectively used for Internet applications. The benefits of using satellite via Internet include [33] high bandwidth, untethered

communication, simple network topology, more manageable network performance, and broadcast/multicast capability. The only disadvantage is the communication latency between two nodes connected by a satellite. We do not need two-way links with the satellite for time synchronization.
The time synchronization information needs to be authenticated. A simple MAC using a shared key could serve the purpose. Encryption or signature with a private key can also be used.

The following are the prime issues that need to be taken care of:

- The communication latency
- The cost of having a receiving antenna in each MANET node
- Weather disturbances that render the system inoperable.

If on the other hand, we had an intermediate network between the satellite and the MANET, then the cost of a receiving antenna as well as the probability of failure due to weather conditions might be reduced.

3.3.2 GPS

The GPS satellite system consists of 24 satellites in 10,000-mile orbits, each circling the earth twice a day and broadcasting radio navigation signals. GPS provides specially coded satellite signals that can be processed in a GPS receiver, enabling the receiver to compute position, velocity and time. Four GPS satellite

signals are used to compute positions in three dimensions and the time offset in the receiver clock. GPS provides timing accuracy in the 300ns range. Some well-designed GPS clocks can remain accurate with only one satellite in view. *GPSClock-200* is a GPS receiver, which is optimized for computing the exact time. It can keep time accurate to a few milliseconds of a second [34, 32].

Benefits – The benefits of GPS include high accuracy, high reliability, worldwide access, precise time worldwide (system wide), reduced calibration cost, small size, low power, low installation cost and low unit cost.

Issues – The satellite transmission for GPS requires a line of sight between the receiver and the satellite. In fact, any bounced signals would just cause erroneous readings, as the signal will be delayed by the additional distance it has to travel, and the delay will throw off subsequent computations. Noise, bias and blunders are other issues to be taken into consideration. Weather conditions also affect the availability of the service. Installing a GPS unit in some of the small, low power devices commonly used in MANETs could be expensive in terms of cost of the natenna and GPS unit.

3.3.3 Overlay of Unmanned Aerial Vehicles

Tactical MANETs are inherently hierarchical and hybrid in nature. In order to minimize the cost of having a uni-directional link with a satellite, we could exploit the underlying hierarchy of tactical MANETs to our advantage. In a

tactical set-up, it is common to deploy one or more Unmanned Aerial Vehicles(UAVs) above the MANET as an overlay depending on the area covered by each UAV. The UAVs are high-power, high memory nodes. All nodes in the MANET are equipped with receiving antennas. Thus, we can use distributed and secure timeservers (UAVs) to perform the required loose clock synchronization by indirect methods. The hierarchy of synchronization servers (overlay network) ensures that only the maximum errors need to propagate. This is somewhat analogous to the base stations in μTesla. This scheme is thus scalable and suitable for multicast, enabling easy joining-member time synchronization without the new member (receiver) needing to send any messages to the sender.

For most cases, a single UAV can cover the entire MANET area. If need be, one can deploy more UAVs to span the entire network, in which case, the routing between the UAVs is bi-directional. The number of UAVs is small (usually one or two), and the mobility of UAVs is not very high. Hence, routing between UAVs can be done effectively by using a pro-active protocol that is efficient for a small set of nodes. We make the following assumptions about the UAVs. They have high storage, high energy and high computational power. Routing between UAVs is bi-directional and effective. The UAVs are trusted entities and do not turn malicious. We also assume that these vehicles are monitored and that their compromise is detected. The communication between UAVs is secure. Each UAV has a common key for all the valid nodes of the MANET. This common key is needed for UAV authentication. We assume that the UAVs are synchronized at

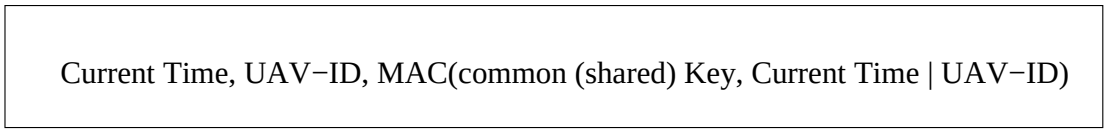


Figure 3.1: Format of UAV Broadcast

all times with each other by a satellite or GPS or other means. The UAVs send timing information in the format shown in Figure 3.1. The MANET nodes can authenticate the packet by verifying the MAC with the UAVs' key, common to all nodes. It is easier to have a common key shared with all the nodes so that the UAV can broadcast the timing information in one go in a bandwidth effective manner. There are no added issues when a node moves from the footprint of one UAV to another. Since it shares the same key with all the UAVs, it will be able to authenticate the next timing broadcast that it receives from its current UAV.

Alternatively, the UAV can append its signature computed using its secret key to the packet. Any node in the UAVs footprint can verify the signature using the UAVs' key. An adversary cannot masquerade as a UAV.

We can use GPS for time synchronization in the overlay network. The accuracy achieved here is many orders more than what is required for the authentication scheme. Hence, the MANET nodes can synchronize themselves with the timing information received from the overlay network. The frequency of overlay broadcast should be adjusted so as keep the time synchronization error between two receivers lesser than that tolerated by the authentication scheme. All the nodes need not have GPS-enabled receiving antennas. The cost is also lesser.

The overlay of UAVs can be used to bootstrap the MANET nodes in an effective manner. This is discussed in the next chapter.

Chapter 4

Applying Tesla/μTesla to the overlay equipped tactical MANET

We propose Tesla and μTesla as candidate algorithms for the MANET. In this work, we evaluate the basic Tesla scheme along with some modifications incorporated from μTesla. In this chapter, we discuss the assumptions made, describe bootstrapping in the presence of overlay network and the tasks that the sender and receiver have to perform. The next chapter discusses the simulation details and experimental results.

4.1 Assumptions

We make the following network assumptions. We assume the presence and availability of an overlay network of Unmanned Aerial vehicles (UAVs). Most tactical networks are hierarchical in nature. The overlay node need not necessarily be a UAV. For example. tanks can also act as overlay nodes. Since tactical MANETs are hierarchical and have overlays, we are not specially

deploying an overlay network just for time synchronization purposes. We also assume that the UAVs are equipped with highly accurate GPS clocks. The number of UAVs is decided based on the network area and on the footprint area of the UAV. In most cases, a single UAV can span the entire network area. We assume high storage and computation facilities in the UAV, in particular - we assume that a UAV can store a symmetric key for every other node in the MANET. The overlay node(s) shares a symmetric secret key individually with each mobile node. The overlay node also has a single common key that it shares with all the MANET nodes.

In order to provide effective time synchronization service to the MANET and to fully exploit the power, memory and other facilities available to the UAV-overlay, there should be a return path from a node in the MANET to the UAV. We need some nodes (*super-nodes*) with two-way links that can send to and receive from the UAVs. We can deploy one or more high power nodes that have two-way links. This way, the ordinary nodes can contact the UAVs through the high-power super-nodes when they need some information. However, for the purposes of time synchronization, we do not need such nodes. Their presence will help us exploit the overlay for other purposes.

4.2 Sender Setup

The sender set up is exactly the same as in [26]. The sender splits time into intervals of length T_{int}, the first interval I_0, starting at time T_0. The sender determines the sending duration and the length of the key chain, N. Each sender first generates a sequence of secret keys (key chain) by choosing the last key K_n randomly. It generates the remaining keys by successively applying a one-way, collision resistant, strong cryptographic hash function, F. Thus,

$$K_i = F(K_{i+1}) \qquad \text{for } i = 1 \ldots N$$

$$K_i = F^{N-i}(K_N) \quad \text{where } F^i(x) = F^{i-1}(F(x)) \text{ and } F^0(x) = x$$

The sender associates each key of the key chain with one time interval and discloses the current key after a delay of the order of a few time intervals after the end of the current time interval. For all packets sent in a given interval, i, the sender uses key $K'_i = F'(K_i)$ for computing the MAC of the data in the packet. This is done to avoid using the same key for commitment and MAC computation. Key chain extensions and use of multiple authentication chains have been discussed in detail in [17, 18, 26]. The sender now needs to bootstrap its receivers in an authenticated manner.

4.3 Bootstrapping

Each receiver needs to be bootstrapped and given one authentic key of the one-way key chain as a commitment to the entire key chain for a particular sender. The receiver also needs to know the key disclosure schedule of the sender. Methods for bootstrapping have been discussed in [17, 18, 26]. Since ad hoc networks are conceived on the fly and the senders are not likely to be known apriori, pre-loading bootstrap information is not appropriate for ad hoc networks. We thus have to resort to secret key or signature based schemes for bootstrapping.

Tesla uses a regular signature scheme to sign the bootstrap information. This is very expensive for resource constrained nodes. μTesla (SPINS) [19] uses the node-to-base-station authenticated channel to bootstrap the authenticated broadcast. The expensive signature is avoided, but it totally relies on the layer above for bootstrapping. It also involves two-way communication between every node and BS. In Ariadne [16], each node is assumed to have a secret key for every other node in the network. The receiver sends a nonce with a request to the sender. Sender replies with its current time, a key of its key chain, the starting time of the interval, the duration of an interval and the disclosure delay. Since confidentiality is not required, there is no need to encrypt the data. The sender replies with the key disclosure schedule, a key chain commitment value and a MAC of these values using the secret key that it shares with the requester

(receiver). This scheme involves exchanges between the sender and the receiver. It is also based on a very strong node assumption about keys. When many receivers try to get bootstrapped to the same sender, there is a bottleneck at the sender and the method does not scale well. In another method in Ariadne, a trusted KDC is used and each node is assumed to share secret keys with the KDC. An alternate protocol given in [16] can do without each node having every other node's key, but involves encrypted message exchange with the KDC. [26] also gives the details of a similar exchange based method.

4.3.1 Solutions

Schemes involving exchanges between the sender and receiver are likely to have a bottleneck at the sender when many receivers try to bootstrap to the same sender and hence do not scale well. Schemes where the sender can multicast the bootstrap packet in an authenticated manner will be efficient in terms of communication overhead. The sender must use a digital signature or keyed MAC for authentication. Digital signatures are expensive in terms of computation and data-overhead. For keyed MACs, the sender needs to use a key that is common to a set of receivers, which is not secure since any receiver can forge packets. Thus, it is evident that the sender and the receiver should share a secret key in order to authenticate the bootstrap packet.

The overlay network can be used to reduce the bottleneck at the sender. As

discussed in Section 4.1 we can often deploy one or more high power nodes (*super-nodes*) with two-way links to the overlay to aid the bootstrap process. In a typical tactical MANET (Figure 1.1), it is common to have such special nodes in the network. The sender should first find a back channel to its parent UAV. This can be "learnt" from the routing information like Hellos and reverse route entries in the routing table. Otherwise, the sender must initiate a "request" and learn the path to the super-node. It can now route the bootstrap packet to the UAV which would then send it to the members of the multicast group by a direct broadcast (if multiple UAVs are present (unlikely), the packet must be sent to other UAVs to reach the receivers that are not in current UAV's footprint). Once the sender registers an initial bootstrap packet with the overlay network, new receivers can be bootstrapped easily by the overlay. Significant work has been done in [35] for hierarchical physical networks and for unidirectional routing.

For authentication, we assume (Section 4.1) that each node in the overlay has a unique shared secret key for every node in the MANET. Since a single UAV can cover the entire MANET area in most cases, we assume that the overlay network consists of a single UAV. Extending it to many UAVs if required is trivial. The sender should use the secret key, K_{S-UAV} that it shares with the UAV to compute the MAC in the bootstrap packet. The UAV can verify the MAC and authenticate the sender since K_{S-UAV} is known only to the sender and the UAV. However, a malicious super-node M could change the sender's identity to its own ID and replace the original sender S's MAC with a new MAC

computed using the key K_{M-UAV}. There is no way the UAV can tell that this packet was tampered with. The new MAC verifies correctly. A keyed MAC would suffice only when all super-nodes are trusted at all times. To circumvent this situation, the sender can encrypt its bootstrap packet with the secret key (that it shares with the overlay nodes) and route it to the overlay through the nearest super-node. The UAV can decrypt the packet and can be assured of the fact that only the node associated with that key could have sent the packet. This method guarantees confidentiality and integrity.

The UAV now needs to broadcast the bootstrap information to the MANET nodes. It uses the common key that it shares with all the MANET nodes to compute the MAC. The UAV then broadcasts this packet to the MANET nodes. If necessary, it can make sure that only the group members can read the contents of the the packet by encrypting it using the associated group's session key. This is somewhat similar to the base station based authenticated broadcast described in [19]. However, any super-node can masquerade as a UAV since it knows the common key and can transmit at the same frequency range as the UAV. The UAV must append a signature computed on the hash of the message before broadcasting it. All the nodes can verify the signature with the UAV's public key. Signature verification is not very heavy on computation as opposed to signature generation. The MANET nodes only need to verify the signatures generated by the high power UAVs. The signature guarantees sender authenticity and unforgeability. All the nodes in a UAV's footprint will be able to

authenticate the packet.

- S $\longrightarrow$ UAV: $\mathcal{E}_{K_{S,UAV}}$(Bootstrap data), where $\mathcal{E}(.)$ is the encryption module
- UAV decrypts packet to get data D
- UAV computes $Sign_{SK}(hash(D))$, where Sign(.) is the signature, SK is the UAV's secret key
- UAV broadcasts data along with signature
- Ground node verifies signature

The receivers can wait for an authenticated broadcast from the overlay before contacting the sender. On the other hand, if the sender is unable to find a path to the super-node and route the bootstrap packet to the overlay, the receivers will not get an authenticated broadcast of the bootstrap packet within the wait-period. The receivers can contact the sender and get the bootstrap packet in the conventional way, i.e. the signature based authentication as described in Tesla or if it is not too expensive for each node to have a shared key with every other node, the nonce-based exchange as used in SPINS. Using the overlay for bootstrapping receivers thus makes the bootstrapping scheme adaptive and more flexible. Once a sender registers an initial bootstrap packet with the overlay network, subsequent receivers can be bootstrapped easily by the UAVs. If the receiver already has a route to the nearest super-node, it can send a request to the UAV through the super-node instead of sending a request to the sender. If, on

the other hand, the receiver has a route to the sender, (or if the route to sender is shorter than route to super-node in case it has routes to both sender and a super-node), it can send a request to the sender in the conventional way. We stress that if sufficient super-nodes are deployed, we need not use the nonce-based exchange or the signature based authentication at all, thus overcoming the shortcomings of bootstrapping mechanisms using in both Tesla and SPINS.

There are other added advantages of having an overlay network. The public-private key pair can be used to send information (like routing information or keys) to a specific node in the network. The UAV acts as a cluster-head with additional memory and storage capabilities. Overlay can be made to detect partitions and provide information to the nodes accordingly.

4.3.1.1 Locating the super-node

Every node should keep a log of the nearest super-node that has the power to contact a UAV. Such super-nodes should broadcast "Hellos" periodically and also as and when they move. The other nodes should keep updating their nearest-super-node info. Forward paths and reverse routes should be set-up to the super-nodes. This has been discussed in detail in [35].

4.4 Receiver tasks

When a node receives a bootstrap packet for a sender, it stores the packet in its buffer after verifying the MAC using the common UAV key or the shared key between the node and the sender, depending on who it receives the bootstrap packet from. The receiver computes the synchronization error with the information in the bootstrap packet. As in Tesla [17, 18, 26], the receiver first verifies the security condition (4.1) for each incoming packet.

$$IntervalID(t_R + Skew^{clk}_{S-R}) \leq IntervalID(t_S) + d \tag{4.1}$$

Here *IntervalID(.)* is a function that assigns the index of the interval (for the keying operation) corresponding to the time argument, t_R is the packet arrival time at the receiver, t_S is the packet transmission time at the sender, d is the disclosure delay (intervals) used in the scheme, and $Skew^{clk}_{S-R}$ is the synchronization difference between the sender and receiver clocks. Only packets that satisfy the security condition are buffered. For every key disclosure packet, irrespective of whether the security condition is satisfied or not, the receiver checks the key authenticity using function F, updates the key commitment and $IntervalID$ corresponding to the latest known key in Tesla. It authenticates all packets sent between the Interval-IDs of the last key disclosure packet and the current key disclosure packet after verifying the MAC. Keys for intermediate intervals are computed by repeated application of the pseudo-random function, F on the latest key.

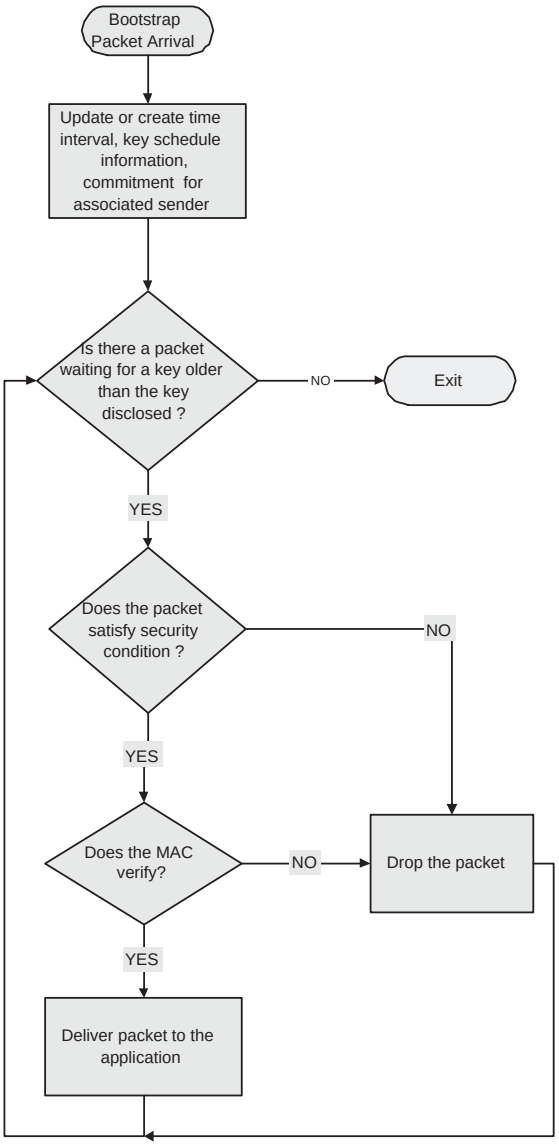


Figure 4.1: Flowchart showing tasks performed at the receiver on receipt of a bootstrap packet: After updating the key-commitment and sending schedule of the sender, the receiver authenticates all packets waiting for keys at least as old as that disclosed.

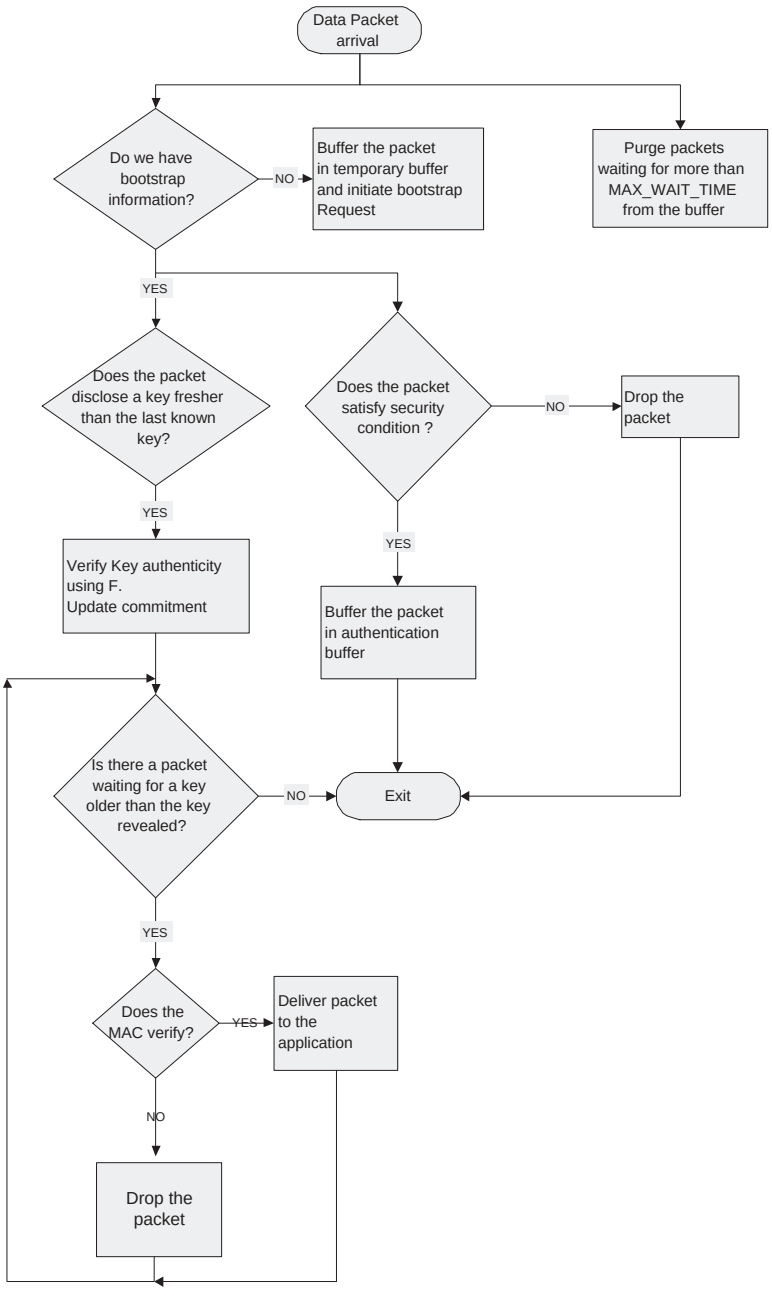


Figure 4.2: Flowchart showing tasks performed at the receiver on receipt of a Tesla packet.

If on joining a group, a receiver receives packets being sent to the group but is unable to authenticate them due to lack of bootstrap info, it must buffer the packets and establish a return path to the current UAV through the nearest super-node and request the UAV to supply bootstrap packets for the sender(s) whose data it wishes to authenticate. If the receiver does not get any bootstrap packets within a certain timeout interval, it must contact the sender S. If the receiver still does not get any bootstrap packet for a certain reasonably large maximum wait time (usually of the order of buffer-timeout), it retries sending requests to the sender(or UAV). After trying for a predetermined number of times, it simply discards packets from its buffer for the corresponding sender.

Chapter 5

Performance Evaluation

5.1 Simulation Framework

We used *ns-2* [20] discrete event simulator with the CMU wireless extensions [36] for our simulations. The IEEE 802.11 Distributed Coordination Function (DCF) [37] as implemented in *ns-2* was used as the Medium Access Control protocol. The radio propagation distance at the physical layer was set to $200m$ for each node. The channel capacity was $1Mbps$. The communication medium is broadcast and nodes have bi-directional connectivity. Our model does not support radio capture [38] and packets are dropped in case of collisions. The *Two-Ray Ground Propagation Model* [39] was used as the underlying propagation model in the physical layer. According to this model,

$$P_r = \frac{(P_t \cdot G_t \cdot G_r \cdot h_t^2 \cdot h_r^2)}{(d^4)} \tag{5.1}$$

where P_t is the transmission power, P_r is the reception power, G_t is the transmitter antenna gain, G_r is the receiver antenna gain, h_t is the height of the transmitter antenna, h_r is the height of the receiver antenna and D is the

MANET Area	1000m x 1000m	TSYN_TIMEOUT	0.5 seconds
Node Tx Range	200m	MAX_TSYN_TRIALS	10
Group Size	50 nodes	Signature Generation Time	10 ms
MAX_WAIT_TIME	5.0 seconds	Signature Verification Time	1 ms

Table 5.1: Simulation Set-Up

distance between the receiver and transmitter. Nodes were randomly placed in an area of $1000m \times 1000m$. Since nodes were randomly placed, network partitions can exist irrespective of the denseness of the network. For all our simulations, we have one multicast group of fifty nodes.

We used MAODV [9] as the multicast routing protocol. The agent was implemented as an application above the network layer so that the data packets with a multicast address as the destination use MAODV as the routing protocol and the unicast packets use the AODV [40] unicast route table. We used the reference implementation of [41]. The MAODV-agent is a derived class of AODV-agent class. We modified the forwarding routines in the code to support broadcast instead of doing multiple unicasts to all activated next-hops in the multicast routing table. The MAODV draft [42] mentions two ways of doing the forward – multiple activated next-hop unicasts or a single broadcast. The implementation in [41] had the former and we modified the code to support broadcast-based forwarding. In a wireless, ad hoc environment, it does not make sense to unicast a packet to each of the activated next-hops. The source becomes

a bottleneck with high sending rates. To effectively test and measure the performance of Tesla, we choose higher sending rates than usual. In a broadcast setting, all nodes in range receive the packet and those that are tree members forward the packet.

Each run simulates two methods to synchronize time – *direct time synchronization* (DTS) and *Indirect time synchronization* (UAV-based).

• For the DTS case, each node sends a unicast request along with a nonce to the source (sender) of the data packet. The sender returns a packet with its sending schedule, key commitment and timing information. The sender signs its reply using its secret key. The receiver verifies the signature and gets time synchronization information. The dispersion is estimated as the difference between the time of arrival of time synchronization request at the sender and the time of despatch of request at the receiver (requester). After sending a request, the receiver sets a timer and waits for TSYN_TIMEOUT period. If it does not receive a reply before this timer expires, it sends another request to the sender and resets the timer. Sequence numbers are used to detect replays. A receiver sends up to MAX_TSYN_REQUESTS to the sender.

• For the UAV case, we assume that all the nodes have bootstrap information and that the UAV periodically broadcasts accurate time. We did so for ease of implementation. This is an ideal case. For both the DTS and the UAV case, the MANET nodes are modeled to have a clock drift of the order of 1 μ-second. The UAV latency is of the order of 40 μ-seconds for current-day UAVs. Thus, the

dispersion with the *UAV's GPS-clock* is negligible for the purpose of our simulations. So, we modeled the node to have a clock with a dispersion of the order of 1 ms from the system clock. This is a pessimistic estimate. Also, in the current simulations, all nodes join the group by the time the sender(s) start multicasting data packets. So, in essence, all nodes wait for the same time for the UAV broadcast. Thus without loss of generality, we can assume that the time at a node is system time, add or subtract a uniformly distributed random value less than 0.001 seconds.

Tesla was implemented above the network layer with both the direct and indirect time synchronization schemes. Each sender is a CBR-generator attached to a UDP-agent. The sender sends a fixed number of packets per second. On receipt of a new data packet, irrespective of whether or not it has time synchronization information, the receiver removes all packets that have waited for more than MAX_WAIT_TIME. The MAX_WAIT_TIME is set to a reasonably high value and packets that receive the key after this time are not delivered to the application layer anyway. The values of specific variables, the MANET area and other specifications of the simulation set-up are given in Table 5.1.

The authentication header of the data payload consists of the key disclosure bit, interval-ID, sequence number and the MAC of the header and the data in the packet. The packet header for DTS request packet consists of the source address and nonce (or sequence number). The reply packet consists of the source and destination addresses, sending time, reply sequence number and signature.

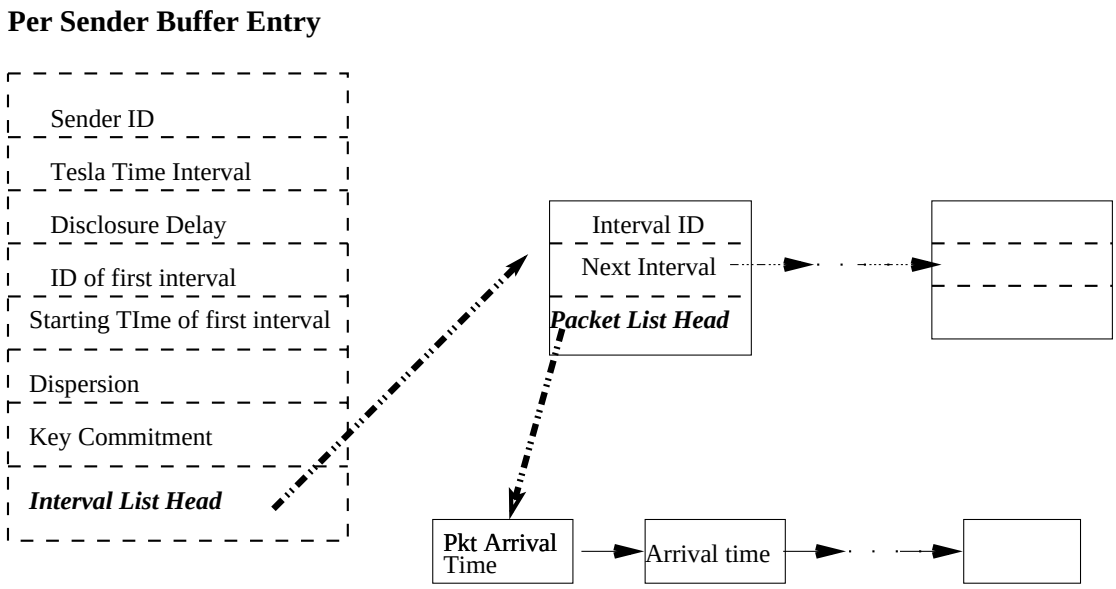


Figure 5.1: Data Structure for Implementing Buffer at Receiver Side

A receiver maintains a buffer to store unauthenticated data packets while waiting for the key. The *authentication buffer* has a structure called *minibuff* for each sender. Each *minibuff* entry has a sender-id, a flag indicating availability of bootstrap information, the last known key for the sender, the interval schedule of the sender consisting of the first interval's id, the starting time of this interval, the Tesla time interval and disclosure delay of the sender, and most importantly, the dispersion between the sender's clock and its own. There is a pointer, *interval list head* to the head of the interval list – a linked list containing the interval-id, a pointer to a list of packet arrival times, and a pointer to the next entry corresponding to the next interval in which packets were received. The packet-list contains the arrival times of incoming packets. The structure is as shown in the

Figure 5.1.

For the direct synchronization scheme, the receiver has another buffer where it buffers packets until it gets time synchronization and bootstrap information. When a receiver gets the required information, all packets in this buffer are first validated and moved to the main buffer if the security condition (4.1) is satisfied, otherwise they are dropped. Also packets that have waited in the buffer for more than MAX_WAIT_TIME are dropped. By purging the buffer every MAX_WAIT_TIME seconds, we thus maintain a buffer whose dimension is limited by time instead of memory space. In a period of MAX_WAIT_TIME seconds, a node can receive at most $(SendingRate \times MAX\_WAIT\_TIME)$ packets.

The *Tesla time interval* and *disclosure delay* were varied as simulation parameters. We performed simulations for both static and mobile scenarios. The simulation parameters were different for both the cases.

• For the static case, we varied packet size, packet generation rate, number of senders (sources). Details of the simulations for the static case are discussed in Section 5.3. Since in the static case, the multicast tree once established, remains stable, our network consisted of 50 nodes, all of which were group members. We did so for the sake of simplicity and to reduce the random seeds involved in our experiments.

• While for the mobile case, we had a fixed packet size, a fixed sending rate and one source. Our network consisted of 75 nodes, 50 of which were randomly chosen

as group members. We used the *random way-point mobility model* [5]. The details of the model as well as other simulation parameters are described in Section 5.4.

5.2 Performance Metrics

We define the following metrics to evaluate the performance of the source authentication scheme.

- Percentage Buffered (*%Buffered*) – is defined as the ratio of the number of packets received by the group members (receivers) that are buffered. As described in Section 4.4, each receiver checks the *security condition* (4.1). The packet is dropped if the security condition is violated. Else, it is buffered and waits for the key. *%Buff* is thus a measure of the fraction of incoming packets that satisfy the security condition. This mainly depends on the routing delay and availability of bootstrap information.

- Percentage Authenticated (*%Authenticated*) – is defined as the percentage of the number of packets received by the group members (receivers) that are authenticated and delivered to the application. On receipt of a packet that discloses a key, irrespective of whether or not it satisfies the security condition, the receiver checks if the key revealed is fresher (newer, i.e. interval id corresponding to the disclosed key is greater) than the last known key for the associated sender. If the key revealed is fresher, the receiver first verifies the authenticity of the key by repeated application of

the pseudo-random function, F as many times as the difference in interval id's between the last known key and the newly disclosed key. It then checks the buffer for packets waiting on this key and authenticates or drops them depending on the outcome of the MAC-check. *%Authenticated* is thus a measure of the fraction of incoming packets that are delivered to the application. This mainly depends on the network conditions. Apart from routing delays, packets can be dropped due to non-availability of bootstrap information. Request implosion at sender could also affect the MAODV good-put.

- Percentage Dropped *(%Dropped)* – is defined as the percentage of packets received that are dropped. Packets are dropped in three cases – when the security condition is violated on receipt, when the packet has been in the buffer for MAX_WAIT_TIME and when the MAC does not verify correctly. $\%Dropped = 100 - \%Authenticated$. This metric does not give any additional information, but is convenient to compare and contrast in many cases.

- Average Buffer Time or Authentication Delay – is the delay prior to authentication. The time elapsed from the time of receipt of the packet to the time of delivery to upper layer is the authentication delay. Only packets that are delivered to the application layer after authentication count for this measure.

5.3 Simulation Results for static case

In this section, we present the results from various simulations performed with static scenarios. We first studied the performance of Tesla on a static scenario. In what we call the " base runs", our network consisted of a single group of 50 nodes with one Constant Bit Rate (CBR) source. The size of each data packet was varied as a simulation parameter for the static case (64,128,256 bytes). The number of such packets generated by each source was also varied (5,10,20 packets per second). Source nodes start generating data at random instants of time. The CBR source sends data packets to the multicast group address. This was done to get a feel of the data delivery rate and routing delay associated with MAODV and to assess the performance of Tesla in such settings. In our model, when a source initially wants to send data, it instructs the routing algorithm to establish the route to the multicast group address. The number of sources was one for the base-runs. We also performed simulations with multiple sources (1, 2, 3, 4, 5) and compared the performance in each case. While performing simulations for multiple sources, the sources generate 64-byte packets at the rate of ten packets per second (5.12 Kbps). Each simulation was run for 250 seconds. For each tuple *(packet size, data rate @ sender)*, simulations were run for time intervals of $0.5, 0.1, 0.25, 0.5$ seconds and disclosure delays of $1, 2, 3, 4, 5$ intervals. Table 5.2 summarizes the simulation parameters for the static case.

On each graph, UAV denotes the indirect time synchronization scheme and

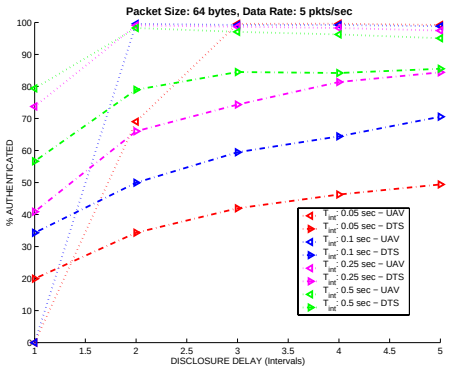
Simulation Time	250 seconds
Packet Size:	64,128,256 bytes
Sending Rate:	5,10,20 pkts/sec
No. of Senders:	1,2,3,4,5
Time Interval:	0.05, 0.1, 0.25, 0.5 sec
Disclosure Delay:	1,2,3,4,5 intervals

Table 5.2: Simulation Parameters for Static Case

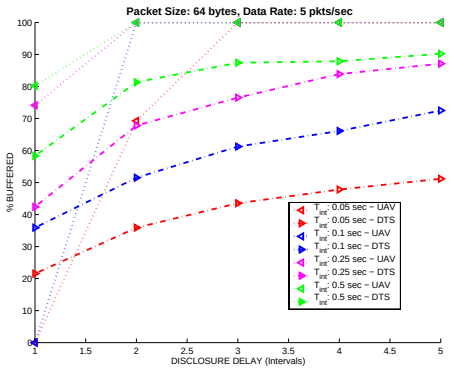
DTS denotes the direct time synchronization scheme. Both schemes were simulated as described in Section 5.1.

5.3.1 Simulations with Single Source

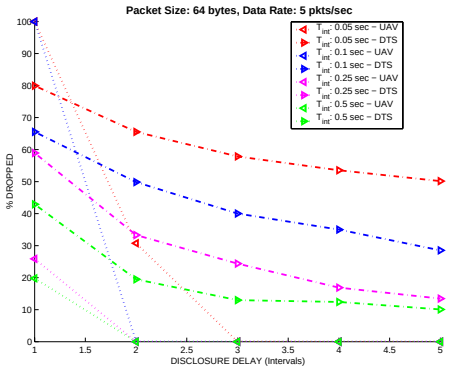
The plots in Figure 5.2 for 64-byte packets sent at the rate of 5 packets/sec show the *%Authenticated, %Buffered, %Dropped* and the *Average Buffer Time* vs. *Disclosure Delay* for different values of the Tesla time interval for both the UAV and the DTS-cases. The bigger the time interval, the longer it takes to authenticate the packet and hence greater the buffer time or the authentication delay. It also means that the probability of the security condition being violated decreases as the interval size increases. When the disclosure delay is 1, no packet satisfies the security condition on receipt if the end-to-end delay is greater than the time interval. This happens when the time interval is 0.05 and 0.1 seconds. Also, since only 5 packets are sent every second and the idea is to use the same



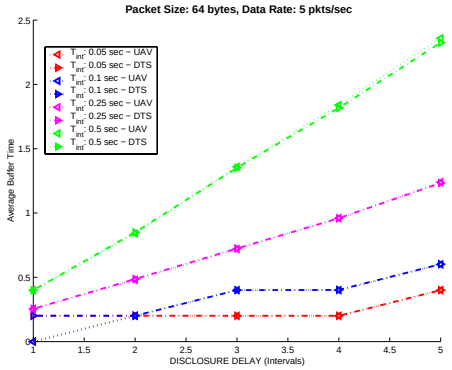
(a) Percentage Authenticated



(b) Percentage Buffered



(c) Percentage Dropped

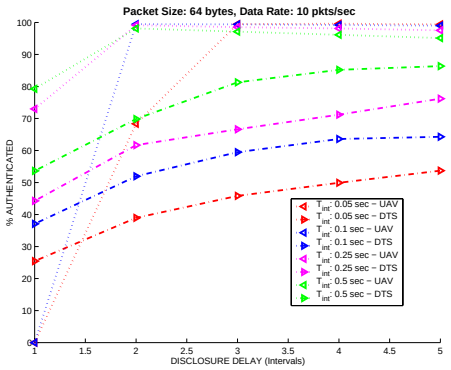


(d) Average Buffer Time

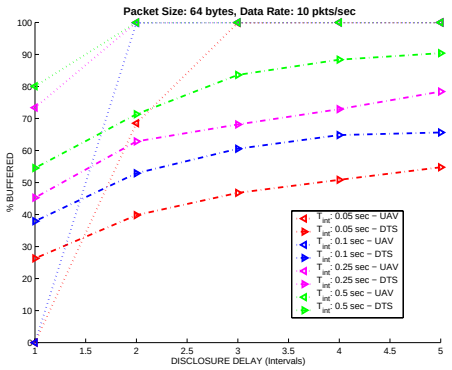
Figure 5.2: Performance Evaluation for 50 nodes, 1 source, packet size 64 bytes, data rate 5 packets/sec

key to authenticate multiple packets (in one time interval), time interval values lower than the sending rate are not "practical" from a key-usage and buffer-maintenance point of view. The UAV case performs better in all cases than the DTS cases. The performance is about 30% more on an average. With a sending rate of 5 packets/sec in a static scenario, the throughput of MAODV is high and almost all packets that are buffered get authenticated as well. This is an indication of the fact that most key-disclosure packets are received by the group members.

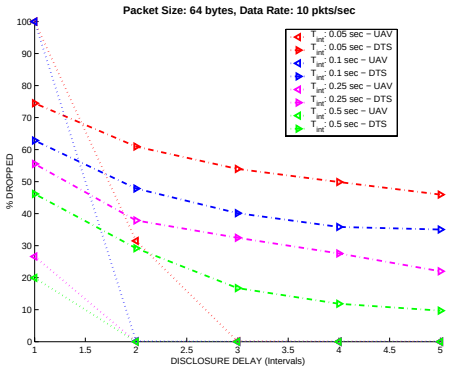
The plots in Figure 5.3, Figure 5.4 show the performance metrics for 64-byte packets sent at the rate of 10 and 20 packets/sec respectively. As in the previous case, the bigger the time interval, the longer it takes to authenticate the packet and hence lesser packet drops due to violation of security condition (4.1) and a larger buffer time. When the Disclosure delay is 1, no packet satisfies the security condition on receipt if the end-to-end delay is greater than the time interval. As before, the UAV case performs better in all cases than the DTS cases. With a sending rate of 10 or 20 packets/sec in a static scenario, the traffic is heavy and the congestion is high. The packets get delayed and hence the number of packets that get buffered after validating the security condition drops slightly. The three sets of graphs for different sending rates are rather similar in nature and nothing concrete can be concluded about the effect of the variation in sending rates on the performance metrics. The average buffer time is almost linear with disclosure delay for a specific value on time interval. The buffer time actually depends on



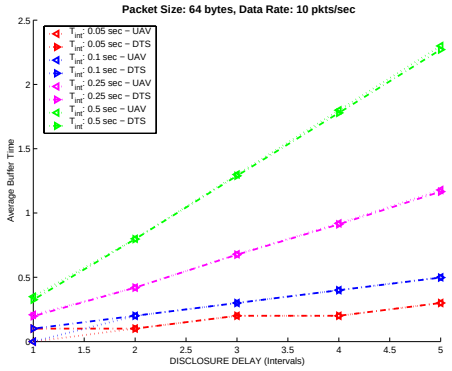
(a) Percentage Authenticated



(b) Percentage Buffered

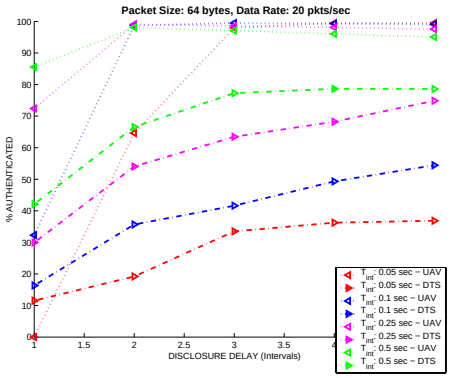


(c) Percentage Dropped

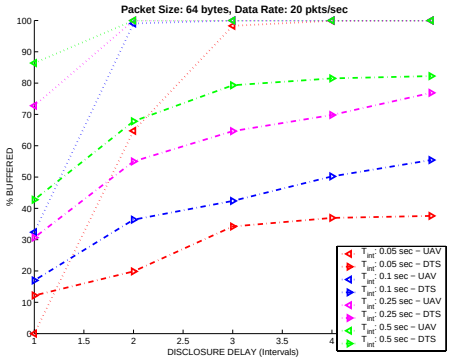


(d) Average MAODV Delay

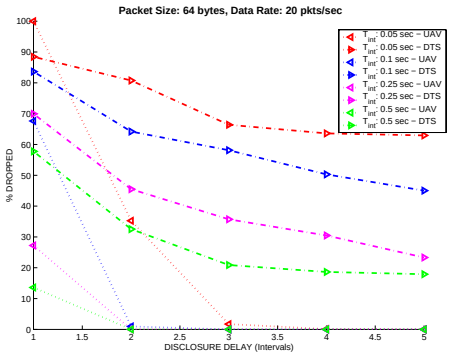
Figure 5.3: Performance Evaluation for 50 nodes, 1 source, packet size 64 bytes, data rate 10 packets/sec



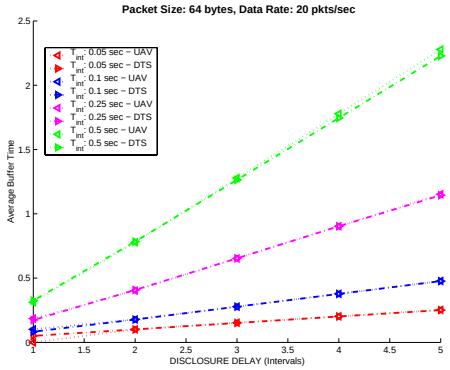
(a) Percentage Authenticated



(b) Percentage Buffered



(c) Percentage Dropped



(d) Average Buffer Time

Figure 5.4: Performance Evaluation for 50 nodes, 1 source, packet size 64 bytes, data rate 20 packets/sec

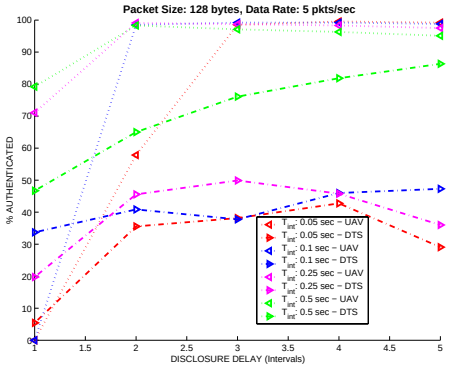
the product of the time interval and disclosure delay.

Figure 5.5, Figure 5.6, Figure 5.7 show the variation of the four metrics with disclosure delay for different time intervals with a packet size of 128-bytes and sending rates of $5, 10, 20$ packets per second respectively. The percentage of packets authenticated drops as compared to the previous case where the packet size was 64-bytes. As mentioned before, the time interval value of 0.05 seconds has no practical significance when the sending rate is less than 20-packets/sec. Similarly, 0.1-second time interval is not relevant for sending rates less than 10 packets per second and so on for other values. There is no fixed pattern among the metrics that vary for all values of the time interval.

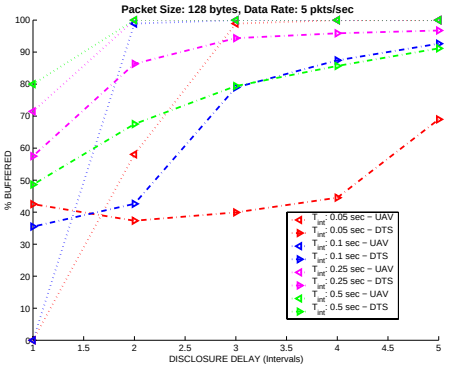
Figure 5.8, Figure 5.9, Figure 5.10 show the performance of the authentication scheme for packet size of $256 - bytes$. When compared to the set of graphs for 64-byte packets, the performance is significantly poorer in this case. The Percentage of packets buffered and authenticated is lesser and the number dropped is higher. The buffer time remains roughly the same as it depends primarily on the $d \times T_{int}$ product, i.e. the absolute time of key-release after sending the packet.

5.3.2 Simulations with Multiple Sources

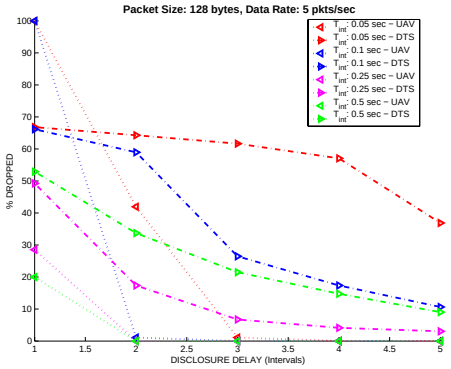
In the second phase of our simulations, we evaluated the authentication scheme coupled with the time synchronization methods on a 50-node group with



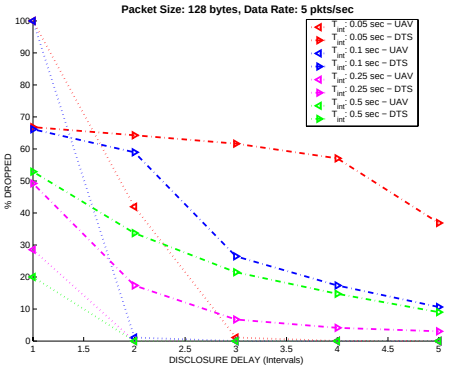
(a) Percentage Authenticated



(b) Percentage Buffered

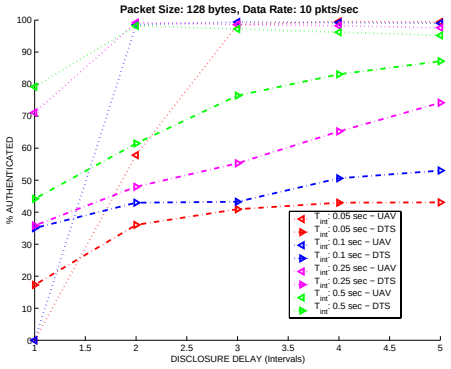


(c) Percentage Dropped

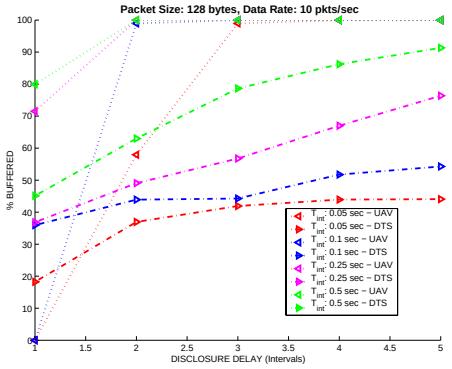


(d) Average Buffer Time

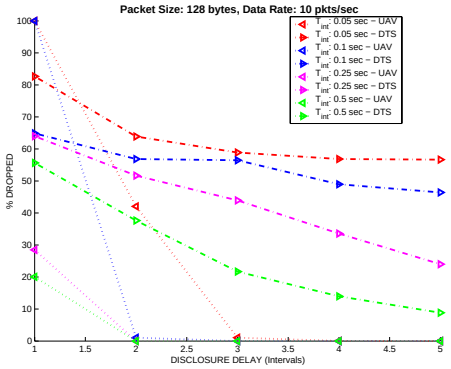
Figure 5.5: Performance Evaluation for 50 nodes, 1 source, packet size 128 bytes, data rate 5 packets/sec



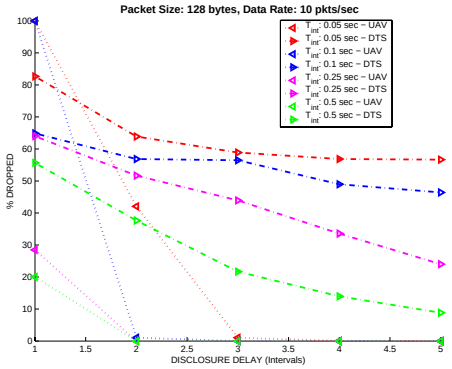
(a) Percentage Authenticated



(b) Percentage Buffered

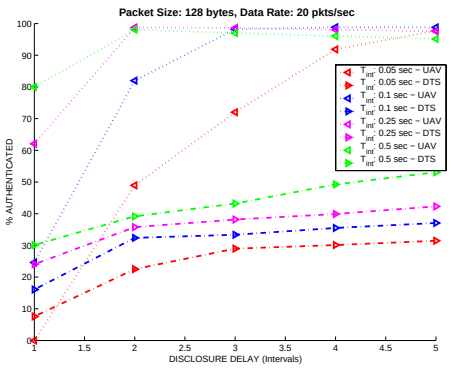


(c) Percentage Dropped

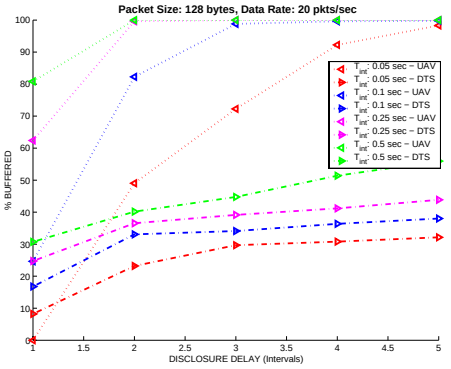


(d) Average Buffer Time

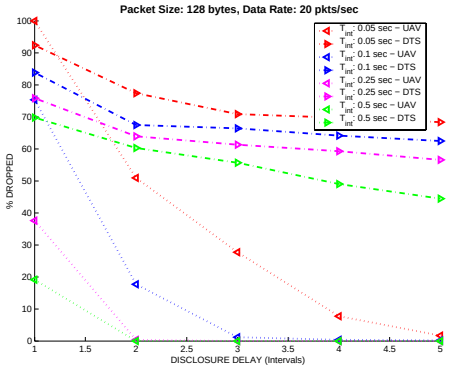
Figure 5.6: Performance Evaluation for 50 nodes, 1 source, packet size 128 bytes, data rate 10 packets/sec



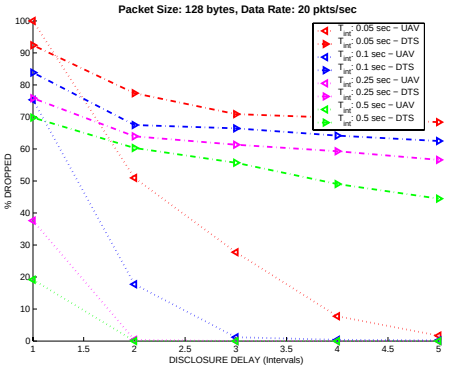
(a) Percentage Authenticated



(b) Percentage Buffered

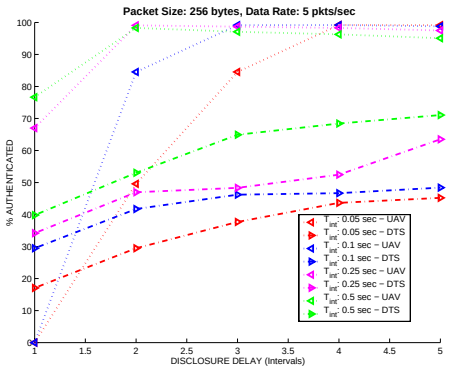


(c) Percentage Dropped

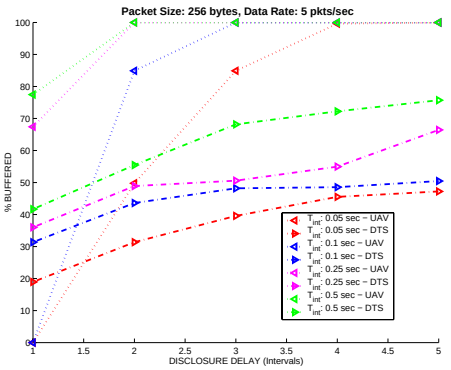


(d) Average Buffer Time

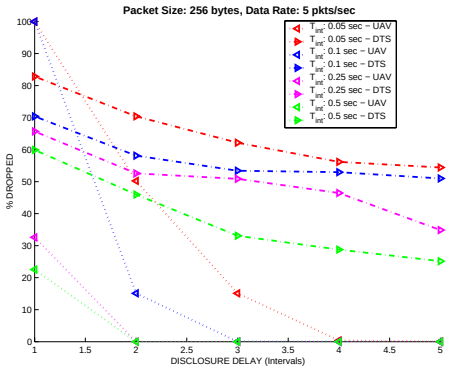
Figure 5.7: Performance Evaluation for 50 nodes, 1 source, packet size 128 bytes, data rate 20 packets/sec



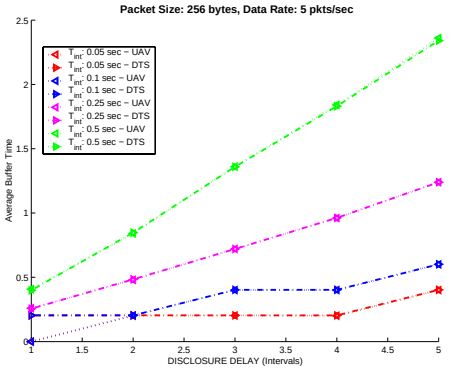
(a) Percentage Authenticated



(b) Percentage Buffered

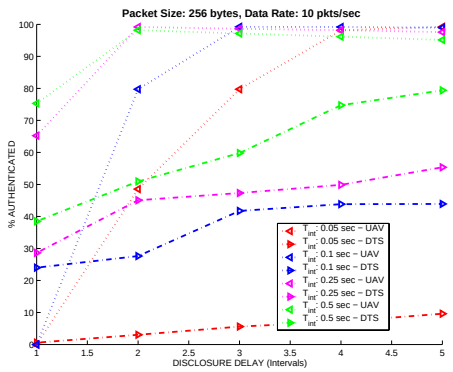


(c) Percentage Dropped

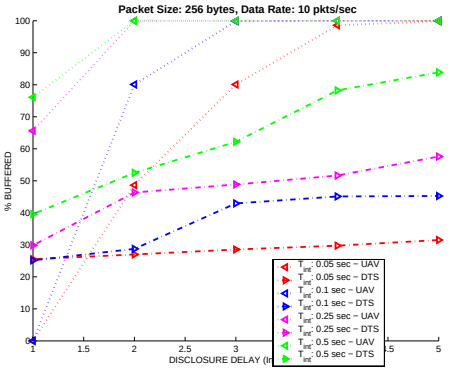


(d) Average Buffer Time

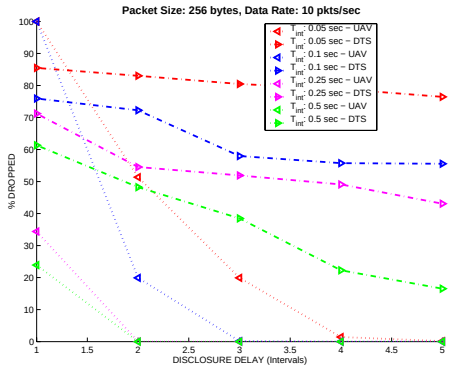
Figure 5.8: Performance Evaluation for 50 nodes, 1 source, packet size 256 bytes, data rate 5 packets/sec



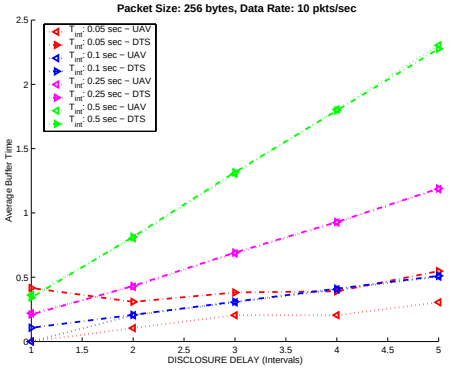
(a) Percentage Authenticated



(b) Percentage Buffered

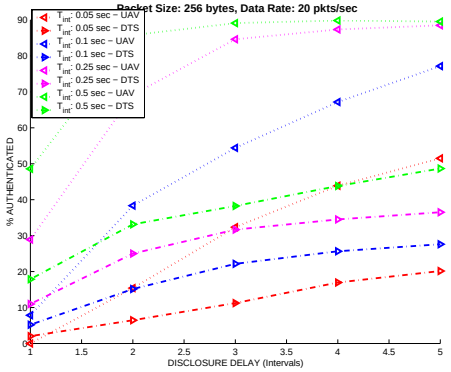


(c) Percentage Dropped

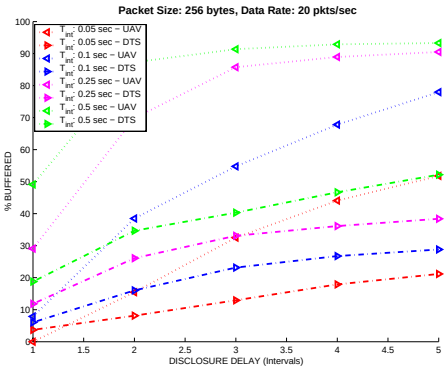


(d) Average Buffer Time

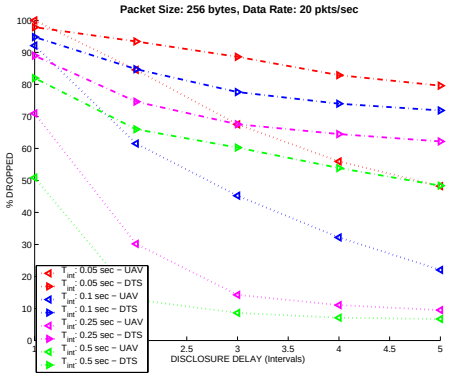
Figure 5.9: Performance Evaluation for 50 nodes, 1 source, packet size 256 bytes, data rate 10 packets/sec



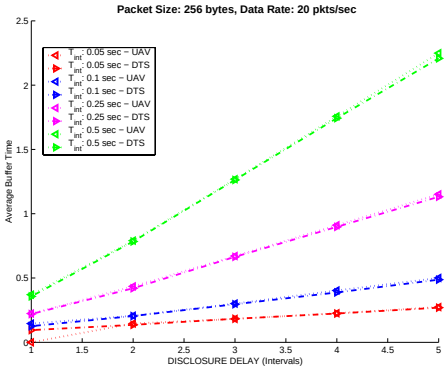
(a) Percentage Authenticated



(b) Percentage Buffered



(c) Percentage Dropped



(d) Average Buffer Time

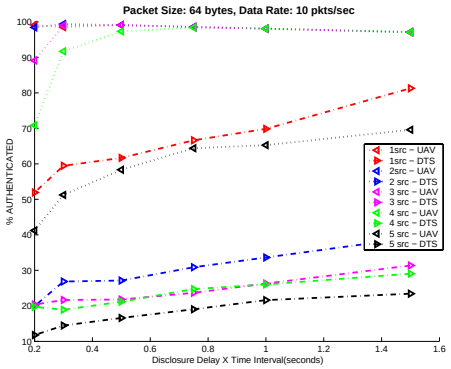
Figure 5.10: Performance Evaluation for 50 nodes, 1 source, packet size 256 bytes, data rate 20 packets/sec

$1, 2, 3, 4$ & 5 senders. Here, the packet size is set to 64 bytes and the senders send packets at the rate of 10-packets/sec. The radio transmission range of a node is 200-meters. The average node degree is about 6.8. The average number of hops from one end of the rectangular area to another is thus 7 hops.

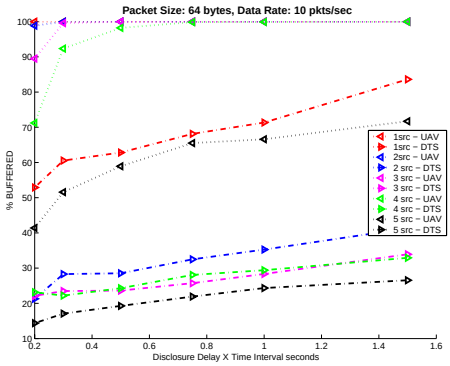
As observed in the earlier plots, the variation of the metrics with the product of time interval and disclosure delay is more starked than with just time interval or disclosure delay. In Figure 5.11, the *%Authenticated, %Buffered, %Dropped, Average Buffer time* are plotted against the absolute time lag in key-disclosure on the sender side, i.e the $d \times T_{int}$ product.

For both the UAV and the DTS schemes, the percentage of packets received that are buffered drops with an increase in number of senders. The congestion in the network increases with an increase in number of senders. The degradation is very obvious for the DTS scheme since the receivers are now trying to find unicast routes to each of the senders and need to get replies back. Due to congestion, a number of these requests and replies get dropped and some receivers may even fail to get synchronized. The overlay network authenticates about 75% more packets than the exchange-based scheme. The average buffer time is roughly the same as the product of disclosure delay and time interval, as predicted from the previous plots.

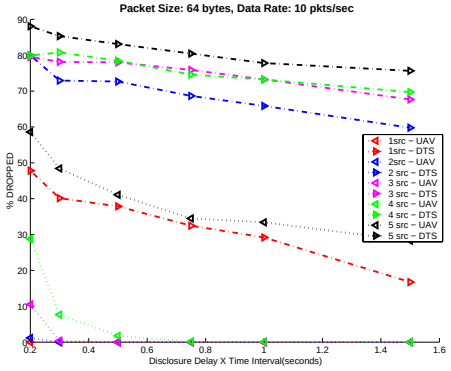
Figure 5.12 shows the MAODV delay and the average clock offset between nodes. In DTS, the dispersion depends on the one-way propagation time of the request from the sender to the receiver. In the presence of an overlay with



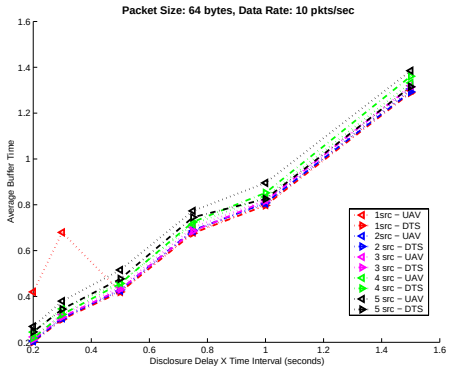
(a) Percentage Authenticated



(b) Percentage Buffered

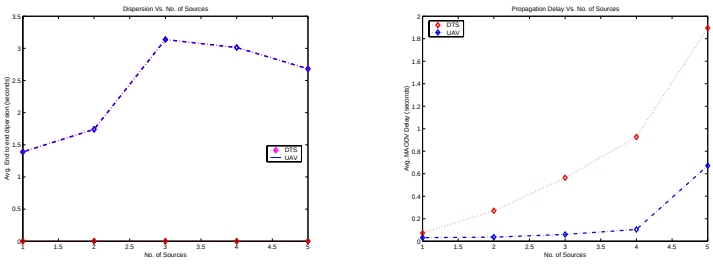


(c) Percentage Dropped



(d) Average MAODV delay

Figure 5.11: Performance Evaluation for 50 nodes, Variable no. of sources, packet size 64 bytes, data rate 10 packets/sec, 200m range



(a) Average clock dispersion (b) Average MAODV Delay

Figure 5.12: MAODV Delay and Clock Dispersion for a multi-hop 50-node group

accurate *GPS-clock*, near perfect clock synchronization is achieved at the cost of an extra receiver antenna and the overlay network.

The average end-to-end MAODV delay for the multicast data payload increases steadily with number of sources for both the overlay and the message exchange-based time synchronization schemes. When the number of senders is 5, there is a lot of congestion in the network since each of the 5 sources send 64-byte packets at the rate of 10 packets per second. Hence the increase in delay. However, in the DTS case, the delay is comparable to that of the UAV case for a single sender scenario. As the number of senders increases, the number of time synchronization requests and replies in the network increases exponentially in the order of the group size. This explains the steep increase in delay for the DTS case. This shows that the direct time synchronization schemes involving message exchanges between the sender and receiver do not scale well and cause delays in routing. The overlay-to-MANET communication is at a different frequency and hence its a separate channel and there is no added contention. For better

performance, it is thus advisable to employ such off-channel methods of bootstrapping and time synchronization.

5.4 Simulation Results for Mobile Scenarios

In this experiment,the mobility speed was varied between 0 and 10 m/s $(0.15, 1, 2, 3, 4, 5, 10)$. The model consisted of 75 nodes randomly placed in the network. 50 multicast group members were randomly chosen from these 75 nodes. We performed our simulations with a single source. The source generated 64-byte packets at the rate of 10 packets/sec.

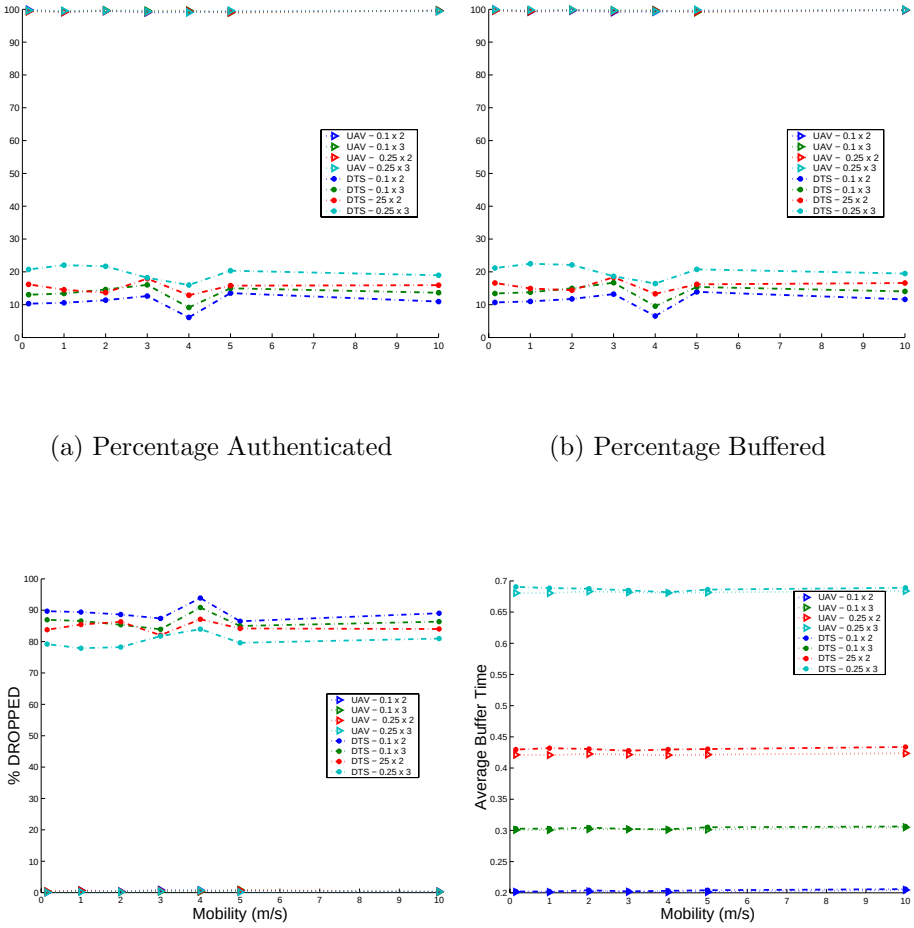
We used the *random way-point model* [5] to generate the movement files for our simulations. In this model, each node selects a random destination in the $1000m \times 1000m$ space and moves to that destination at a speed uniformly distributed between a minimum speed and some maximum speed, v_{max}. Upon reaching that destination, the node pauses for *pause time*, p_t seconds, selects another destination and moves towards the destination as described above, repeating this pattern of movement for the entire duration of the simulation. Thus, the nodes are free to move to any location in the network area. The continuous movement of the nodes ensures continuous change in the topology. This highly dynamic network is ideal to test the tenacity of the message-exchange based time synchronization scheme. We discard the first 3600 seconds while generating the movement scenario files. This is done to "warm-up"

the simulation and avoid some drawbacks of the Random Way-point model. Each simulation was run for 500 seconds. We generated three movement files for each chosen speed. For each of the three scenarios, we collected data over five sets of group selections. The numbers reported are averaged over these 15 runs in order to effectively capture the random choice of group members and movement scenarios. We use a pause time of 50 seconds for our simulations.

MANET Size	75
Group Size	50
Duration	500s
Packet Size:	64 bytes
Sending Rate:	10 pkts/sec
No. of Senders:	1
Time Interval:	0.1, 0.25 seconds
Disclosure Delay:	2,3 intervals
Node Mobility	0.15,1,2,3,4,5,10 m/sec

Table 5.3: Simulation Parameters for Mobile Set-up

Figure 5.13 shows the plots of percentage of packets authenticated, buffered, dropped and the average time spent in the buffer prior to authentication against mobility speed. The UAV case clearly outperforms the DTS method of time synchronization. For the DTS case, there is a slight increase in the percentage of packets authenticated as the $d \times T_{int}$ product increases, where d is the Tesla



(a) Percentage Authenticated

(b) Percentage Buffered

(c) Percentage Dropped

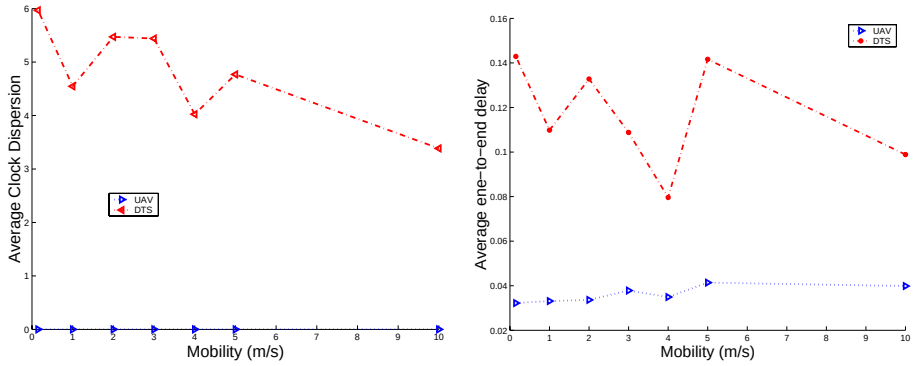
(d) Average Buffer Time

Figure 5.13: Performance Evaluation for a single source, 50-node group with Mobility Speed (0-10 m/s)

disclosure delay and T_{int} is the Tesla time interval. We ran simulations for Tesla time intervals of $0.1, 0.25$ seconds and disclosure delays of 2 and 3 intervals for each case. A similar trend can be seen in the curves in Figure 5.13(b) and Figure 5.13(c). There is no steady decrease in the figures as the speed increases. This is because of probable network partitions and the use of random seeds to generate the movement patterns and also to select the group members among the MANET nodes. These choices vary for different cases and the numbers shown are averaged over 15 runs as explained above. Our goal is to compare the performance of Tesla for the DTS and UAV case rather than performing sensitivity studies of the metric with respect to mobility speed.

The average buffer time depends primarily on the $d \times T_{int}$ product. The delay for the DTS case is marginally higher than the UAV case. This is because of the packets stored in the temporary packet buffer until time synchronization information in available. The packet arrival time is noted and these packets are buffered. The security condition is evaluated only after the time synchronization and bootstrap information are available. This extra time to get the required information is reflected by an additional delay for the packets in the temporary buffer.

Figure 5.14(a) shows the average receiver-to-sender clock dispersion as calculated by the DTS method. The dispersion for the UAV case is the pessimistic estimate of 0.001 seconds, as explained in Section 5.1 in the beginning of this chapter. The dispersion for the DTS case is significantly higher. The



(a) Average clock dispersion (b) Average MAODV Delay

Figure 5.14: MAODV Delay and Clock Dispersion

receiver has to find a route (unicast) to the sender and get the time synchronization information. The scenarios used have an average of 6 to 7 hops. There is no fixed trend with increase in mobility. There may be a partition in the network at a lower speed and none at a higher speed. We use the random way-point model and network conditions may vary from one speed to another. However, it is evident that the high dispersion in the DTS case causes the security condition to be violated more often and hence the poor performance metrics, as shown in Figure 5.13. Figure 5.13(b) shows the average end to end delay for a multicast packet. The delay again does not follow any trend with increasing mobility due to reasons explained above. The delay is more for the DTS case because of increase in CBR traffic in the network due to the flooding of time synchronization requests and replies.

Tesla performs well even in mobile scenarios for the UAV case. Thus, it is

clear that the authentication scheme is suitable for multicast settings in MANETs. The DTS scheme delays the authentication process and causes a lot of packet drops due to violation of security condition and non-availability of time synchronization information. Whereas use of UAV for the same guarantees high performance and operability in all scenarios irrespective of topology changes and node movement.

Chapter 6

Conclusions and Future Work

In this chapter, we first present a brief qualitative security analysis of the chosen authentication scheme. This is followed by the conclusions drawn from simulation results presented in the previous chapter. In the last section, we briefly outline the future work and the motivation for performing the comparisons listed under future work.

6.1 Security Analysis

Tesla guarantees source authentication and message integrity in the following ways:

1. All messages that do not satisfy the *security condition* at the receiver are discarded. The security condition (4.1) makes sure that the key used to compute the MAC of a packet's data portion is known only to the sender at the time of arrival at the receiver. This guarantees that only messages sent by the purported sender are accepted.

2. Packets that satisfy the security condition are buffered and on receipt of the key, the MAC is verified by the receiver. Packets whose MAC does not verify are discarded. This guarantees the integrity of accepted messages.

The security condition thus prevents a packet that was modified in transit from being authenticated. It also prevents an intermediate node from masquerading as the sender. However, a malicious node can copy the "data" portion of the packet and replace the MAC with a MAC using its key. The scheme thus does not provide non-repudiation.

Indirect time synchronization wards off Denial of Service (*DoS*) attacks at the sender. Our overlay architecture prevents a *DoS* attack at the sender. However, if the overlay is used for bootstrapping, there is a possibility of the UAV(s) getting overwhelmed by requests from MANET nodes. Firstly, since MANET nodes have to pass through some special nodes (super-nodes) to pass on a bootstrap-request to the UAV, checks could be added in such nodes to prevent a bottleneck at the UAV. There is a possibility of *DoS* in these super-nodes. Since UAVs are powerful nodes, monitoring and policing can be done with ease. Secondly, as described in Section 4.3, the bootstrapping mechanism is made adaptive by the overlay network. If a node fails to get the required bootstrap information from the overlay, it sends a request to the sender in the conventional way. Thirdly, communication with the UAVs takes place at a different frequency from the MANET communication and hence will not directly affect the network since this

is a separate channel.

DoS at the receiver side can be created in many ways and has been analyzed in detail in [18]. Delayed packets will violate the security condition. Replay packets do not do much harm since a duplicated packet is accepted by the receiver only within a very short time period as the security condition is violated. Receivers reject packets if a malicious node tries to create a *DoS* attack by sending packets marked as being from an interval in the future as the security condition will be violated. Replay can be prevented by adding sequence numbers in the MAC.

However, the scheme cannot prevent a legitimate member from turning malicious and stop forwarding packets. It cannot detect a compromised node. Neither does it prevent a node from generating a false route error message. It does not prevent all *DoS* attacks. Worm-hole detection is also not a problem addressed by the authentication scheme. All these are problems that the routing protocol must handle. We stress that although these routing attacks can affect communication in the system, it will not breach the security of the authentication scheme, i.e. it will not force spurious packets to be authenticated. The security is not compromised. However, the performance is affected since the authentication scheme depends on the timely arrival of the key disclosure packets, bootstrap and time synchronization information, all of which heavily depend on proper functioning of the routing protocol.

We summarize the key points discussed above.

- Authentication, Integrity are guaranteed.
- Confidentiality can be achieved if needed.
- Non-repudiation is not provided.
- *DoS* at sender is eliminated in the UAV case.

  DoS at the receiver is kept in check by using sequence numbers.

  There is a possibility of *DoS* at the UAV and super-node if the overlay is used for bootstrapping.

6.2 Conclusions

From the simulation results presented in the previous chapter, it is evident that Tesla is well-suited for a tactical MANET. Tesla is resistant to collusion and has very low communication and data overhead. It is also resistant to packet loss and can operate in conjunction with any unreliable transport protocol. Generating the authentication information is inexpensive and the overhead is as low as 20 bytes per packet. We also showed that indirect methods of time synchronization fare much better than direct, exchange-based methods. In particular, we compared the performance of the nonce-based query-response method and the overlay-based indirect time synchronization method. It is clear that the UAV-based method is many orders of magnitude more efficient than the direct time synchronization scheme. The DTS scheme degrades the performance of

Tesla in mobile scenarios. The traffic generated by the requests and replies also affect the MAODV end-to-end propagation delay and lesser number of packets are received and authenticated on time. In Chapter 4, we also outlined the other added benefits of the overlay network.

6.3 Future Work

The performance of the authentication scheme is not directly related to the routing protocol used. The performance however depends on timely arrival of packets. A multicast routing protocol that has less delay and communication overhead is likely to enhance the performance of Tesla. We used MAODV, a tree-based on-demand multicast routing protocol for our simulations. Any link breakage triggers actions to repair the tree, making it very sensitive to node mobility.

ODMRP [10] is a popular mesh-based routing protocol that uses a forwarding group to maintain multiple routes between nodes in the multicast group. A link failure need not necessarily mean reconfiguration as in the case of MAODV since there are alternate paths between nodes [43]. ODMRP however does not scale well. Thus, depending on the application, one should first choose the routing scheme for multicast and then tailor the key establishment and authentication mechanisms to perform well in conjunction with the multicast routing protocol. As future work, we plan to study the performance of Tesla on ODMRP. We also

plan to simulate the super-nodes and UAVs for bootstrapping the authentication scheme.

BIBLIOGRAPHY

[1] S. Jacobs and M.S. Corson. *MANET Authentication Architecture*. Internet Draft, IETF, August 1998.

[2] Mobile Ad hoc Networks IETF Chapter. http://www.ietf.org/html.charters/manet-charter.html.

[3] L. Zhou and Z.J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine, special issue on networking security*, 13(6):24–30, Nov,Dec 1999.

[4] J-P. Hubaux, L. Buttyan, and S. Capkun. The Quest for Security in Mobile Ad Hoc Networks. In *Proceedings ACM Symposium on Mobile Ad hoc Networking and Computing (MOBIHOC)*, 2001.

[5] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, 1998.

[6] C.E.Perkins. *ADHOC NETWORKING*. Addison Wesley, 2001.

[7] S.E. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, December 1991.

[8] T. Maufer and C. Semeria. *Introduction to IP Multicast Routing*. Internet Draft, draft-ietf-mboned-intro-multicast-01.txt, March 1997. Work in progress.

[9] E. Royer and C. E. Perkins. Multicast Operation of Ad hoc On-Demand Distance Vector Routing Protocol. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 201–218, Seattle, WA, August 1999.

[10] S.-J. Lee, W. Su, and M. Gerla. *On Demand Multicast Routing Protocol (ODMRP) for Ad hoc Networks*. Internet Draft, draft-ietf-manet-odmrp-01.txt, June 1999. Work in progress.

[11] P. Sinha, R. Sivakumar, and V. Bharghavan. MCEDAR: Multicast core extraction distributed ad hoc routing. In *Proceedings of the Wireless Communications and Networking Conference*, 1999.

[12] M. Liu, R. Talpade, A. McAuley, and E. Bommaiah. AMRoute: Ad hoc Multicast Routing Protocol. Technical Report 8, University of Maryland, 1999.

[13] J. Moy. Multicast Routing Extensions for OSPF. *Communications of the ACM*, August 1994.

[14] M. J. Moyer, J. R. Rao, and P. Rohatgi. A Survey of Security Issues in Multicast Communications. *IEEE Network Magazine*, 13(6):12–23, Nov/Dec 1999.

[15] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast Security: a Taxonomy and Some Efficient Constructions. In *Proceedings of IEEE INFOCOM*, pages 708–716, Mar 1999.

[16] 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, http:www.monarch.cs.rice.edu/monarch-papersariadne.ps, 2001.

[17] A. Perrig, R.Canetti, B. Briscoe, J.D. Tygar, and D.Song. *TESLA: Multicast Source Authentication Transform.* Internet Draft, draft-irtf-smug-tesla-00.txt, Nov 2000. Work in progress.

[18] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient and Secure Source Authentication for Multicast. In *Network and Distributed System Security Symposium (NDSS)*, Feb 2001.

[19] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Proceedings of MobiCom*, Rome, Italy, July 2001.

[20] Network Simulator. http://www.isi.edu/nsnam/ns.

[21] P. Rohatgi. A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication. In *ACM Conference on Computer and Communications Security*, pages 93–100, Nov 1999.

[22] R. Merkle. A Certified Digital Signature . In *Advances in Cryptology*, pages 218–238, Aug 1989.

[23] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. In *Advances in Cryptology*, pages 180–197, Aug 1997.

[24] C. Wong and S. Lam. Digital Signatures for Flows and Multicasts . In *Proc. IEEE ICNP*, Austin, TX, Oct 1998.

[25] T. Hardjono and G. Tsudik. IP Multicast Security: Issues and Directions. In *Proceedings of Annales de Telecom*, 2000.

[26] A. Perrig, R. Canetti, J.D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, May 2000.

[27] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb 1978.

[28] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth*

Symposium on Operating Systems Design and Implementation (OSDI), pages 147–163, Boston, MA, December 2002.

[29] K. Romer. Time Synchronization in Ad Hoc Networks.

[30] M. Bishop. A Security Analysis of the NTP Protocol. www.eecis.udel.edu/ntp/ntpspoo\%l/doc/security.ps.gz.

[31] D. L. Mills. Internet Time Synchronization: The Network Time Protocol. *IEEE Trans. Communications*, 39:1482–1493, Oct 1991.

[32] W. Lewandowski, J.Azoubib, and W.J. Klepczynski. GPS: Primary tool for time transfer. *IEEE Proc.*, 27(1):163–172, 1999.

[33] Y. Zhang, D. DeLucia, B. Ryu, and S. Dao. Satellite Communications in the Global Internet: Issues, Pitfalls, and Potential. In *INET 97*, Malaysia, June 1997. http://www.wins.hrl.com/people/ygz/papers/inet97.html.

[34] GPS Clocks for Computer Time Synchronization. http://www.gpsclock.com/gps.html.

[35] Y-B Ko and N. H. Vaidya. A Routing Protocol for Physically Hierarchical Ad Hoc Networks. Technical Report 97-010, Computer Science, Texas A & M Univ, Sep 1997.

[36] Monarch ns-2 wireless extensions from Monarch Group, Carnegie Mellon University. http://www.monarch.cs.cmu.edu/cmu-ns.html.

[37] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Protocol (MAC) and Physical Layer (PHY) Specification*. IEEE, IEEE Std 802.11 edition, 1997.

[38] C. Ware, T. Wysocki, and J.F. Chicharo. Simulation of Capture Behavior in IEEE 802.11 Radio Modems. *Journal of Telecommunications and Information Theory*, 2001.

[39] T.S. Rappaport. *Wireless Communications: Principles and Practice*, pages 120–126. Prentice Hall, second edition edition, 2002.

[40] C. E. Perkins and E. M. Royer. Ad hoc On-demand Distance Vector (AODV) Routing. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999.

[41] Wireless Multicast Extensions to AODV / DSR in ns-2.1b9a. http://www4.cs.uni-dortmund.de/~Lindemann.

[42] E. Royer and C. E. Perkins. *Multicast Ad hoc On Demand Distance Vector Routing*. Internet Draft, draft-ietf-manet-maodv-00.txt, July 2000. Work in progress.

[43] S-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *INFOCOM*, pages 565–574, 2000.