Mississippi State University
Scholars Junction

Theses and Dissertations | Theses and Dissertations

1-1-2007

A Dual-Agent Approach For Securing Routing Protocols

Brian Lee

Follow this and additional works at: https://scholarsjunction.msstate.edu/td

Recommended Citation
Brian Lee, "A Dual-Agent Approach For Securing Routing Protocols" (2007). *Theses and Dissertations*. 120.
https://scholarsjunction.msstate.edu/td/120

This Graduate Thesis - Open Access is brought to you for free and open access by the Theses and Dissertations at Scholars Junction. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholars Junction. For more information, please contact scholcomm@msstate.libanswers.com.

A DUAL-AGENT APPROACH FOR SECURING ROUTING PROTOCOLS

By

Brian Lee Gaines

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulf llment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science and Engineering

Mississippi State, Mississippi

Dec 2007

Copyright by

Brian Lee Gaines

2007

A DUAL-AGENT APPROACH FOR SECURING ROUTING PROTOCOLS

By

Brian Lee Gaines

Approved:

Mahalingam Ramkumar Assistant Professor of Computer Science and Engineering (Major Professor)	Rayford B. Vaughn Professor of Computer Science and Engineering (Committee Member)
Yoginder Dandass Assistant Professor of Computer Science and Engineering (Committee Member)	Edward B. Allen Associate Professor of Computer Science and Engineering, and Graduate Coordinator

Roger King
Associate Dean
for Research and Graduate Studies
of the Bagley College of Engineering

Name: Brian Lee Gaines

Date of Degree: Dec 14, 2007

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Mahalingam Ramkumar

Title of Study: A DUAL-AGENT APPROACH FOR SECURING ROUTING PROTOCOLS

Pages in Study: 46

Candidate for Degree of Master of Science

Ad hoc routing inherently serves two separate and conf icting divisions of interest: the needs of the user or individual and the needs of the network or community. These interests should be secured differently.

The proposed research is a dual-agent approach for securing ad hoc routing protocols. This approach assumes a physical division of tasks into user agent tasks and tasks performed by a trustworthy network agent. The research, motivated by the need to reduce the tasks of the network agent, investigates strategies for an optimal division of labor while promoting the faithful execution of the routing protocol. This investigation employs the dual agent approach for securing a variant of distance vector routing.

DEDICATION

I dedicate this work to my family and friends. I would not have made it this far without each of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

Bloom f lter duration The range of time associate with a specif c bloom f lter, measured in units of intervals

client A node consisting of a network agent and user agent

interval Or time interval is an arbitrary unit of measurement in which all clients in a network are able to transmit and process routing information among neighbors

network agent The portion of the client that is tamper resistant. The user agent has partial control over the network agent

node A laptop, PDA, sensor, or other wireless device participating in an ad hoc network

user agent The portion of the client that a user has complete control over

CHAPTER 1

INTRODUCTION

The world is covered in networks from end to end: phone, computer, wired and wireless networks. One of the primary services that a network requires is routing. Routing, in its simplest form, is providing a path between two entities in a network and forwarding data along that path for communication purposes. Early routing research completely ignored the possible presence of malicious entities in the network. However, this optimistic view does not hold in practice, malicious entities are present and their ability to disrupt routing must be addressed.

1.1 Problem Statement

In conventional networks the task of routing is performed by large network operators. In mobile ad hoc networks, which do not rely on existing network infrastructure, every node has to cooperate to route packets between them. The tasks performed by a node in an ad hoc network can be classified into "selfish" tasks that are useful only for the node performing the task and "selfless" tasks performed for the benefit of the entire network. Securing such cooperative tasks requires proactive measures to ensure that the nodes will indeed adhere to rules that govern the tasks. Obviously, individuals have more motivation

to abide by the rules that govern self sh tasks. Thus, the design of strategies for ensuring adherence to rules should consider the different approaches for securing different tasks.

One of the primary limitations of existing ad hoc routing protocols stems from the lack of distinction between approaches to protect the integrity of self sh and self ess tasks. This distinction should not only be a logical partition but a physical partition. Recent work [17] has indicated the feasibility of low cost trustworthy computers, as long as the complexity of the trustworthy computer is maintained at very low levels. The need for distinction of tasks and the feasibility of low cost trustworthy computing modules, together, are the motivators for the dual-agent approach for routing protocols.

1.1.1 Research Goals

The division of labor between the agents is considered to f nd a balanced solution that provides both necessary security and minimal requirements on the network agent. The primary goal is to reduce the cost of the network agent by reducing the complexity of its tasks, in terms of its computational and storage requirements. Inherent limitations of the dual-agent approach, primarily arising from the network agent's lack of resources, are considered.

The primary challenge is optimal tasks sharing strategies, while taking into consideration the different levels of protection that need to be extended to different tasks and the inherent limitations of the network agent. The network agent tasks are limited to those that can be performed by a simple logic engine that can eff ciently reuse a single hardware

block cipher. Specifically, this thesis addresses dual agent strategies for securing a variant of a distance vector routing protocol.

1.1.2 Hypothesis

The hypothesis is that an optimal division of labor can be found between the network agent and the user agent that reduces the requirements of the network agent to

- a symmetric block cipher (used for encryption and hashing) and
- a few kilobytes of storage.

Regardless of these limitations, the low complexity network agent will provide assurances of security against different types of attacks by the user agent.

1.1.3 Research Approach

The attacks against the dual agent approach, consisting of strategies of the user agent to circumvent the protocol, were identified and classified into four broad categories: fabrication, selective dropping, omission, and replay attacks. Low complexity strategies (within the capabilities of the network agents) to address these attacks were identified. Specifically, a strategy similar to the next secure (NSEC) record (used in DNSSEC for authenticated denial of existence of DNS resource records) and Bloom filters were identified as efficient strategies to address the attacks.

To obtain an estimate for the overheads required for the network agent tasks in practical deployments, the broad strategies designed to protect the integrity of routing protocols were applied to a distance vector protocol. More specifically, the protocol tested was based

on the Destination-Sequenced Distance-Vector (DSDV) protocol. A simulation environment was developed which implements the dual-agent distance-vector protocol. For the purposes of off ine analysis of potential attacks and complexity of tasks to address such attacks, the environment produces snapshots of routing tables of all nodes were recorded at the beginning of each interval. The environment is also capable of producing a perpetual list of real-time revoked routes.

CHAPTER 2

PRIOR WORK

An early example of routing is the transfer of a phone call across the telephone system. The phone system once used humans as switching units to manually create routes to connect two people. Improvements in technology replaced people with analog switches, and eventually with digital switching equipment. Many of the same technologies are used to connect computers in small networks, interconnect networks, and even support the Internet. Because of this, routing is a signif cant research area. Some of the goals for routing are to provide the fastest connection, the shortest connection, or the least expensive connection between two parties. Routing strategies must also be able to handle dynamic changes in network topology. Different routing strategies may be suited for meeting the requirements unique to different applications.

Ad hoc networks are a more recent concept in networking. These networks typically require the members to implicitly trust one another, and in the very least, they must trust the routing information received. DSDV [16], Ad Hoc on Demand Distance Vector Routing (AODV) [15], and Dynamic Source Routing (DSR) [12, 11] are some examples of ad hoc routing protocols. All such protocols explicitly expect the members of the network to fully cooperate in the protocol. This implicit trust has also proved to be their greatest weakness.

Security was not considered in the initial development of ad hoc networking protocols. This is not surprising since routing protocols are extremely complex.

2.1 Distance Vector Routing

Distance-vector routing has many names: Bellman-Ford routing, Ford-Fulkerson algorithm, RIP, etc. Basic distance-vector routing is a proactive protocol since it relies on periodic updates of routing information. With this type of routing a node is expected to learn the distance value associated with all other nodes in the network. Routing information is passed as routing tables between neighbors. The distance value represents a network measurement, such as hop count. Routing tables consist of node id, best distance value to reach the node, and next hop for reaching the node. Periodically the nodes exchange tables, indicating node id and the best distance value of all nodes, with their neighbors. Following receipt of every routing update, the nodes make appropriate corrections to their tables [22].

One serious f aw of distance-vector routing is the count to inf nity problem[22]. The problem occurs when a node becomes unavailable. Figure 2.1 is a small example network. Once the network has reached steady state *node 10* has a route to *node 7* through *node 4*. Now assume that *node 7* becomes unavailable. The next time *node 10* sends routing information it will include a route to *node 7* of length two. *Node 4* no longer has a route of length one to *node 7*, and because of this it accepts the route of length two from *node 10*. Now *node 4* has a route to *node 7* with the next hop as *node 10*, and *Node 10* has a route to

node 7 with a next hop set to *node 4*. The circular route continues each update increasing the hop count until it reaches some maximum value.

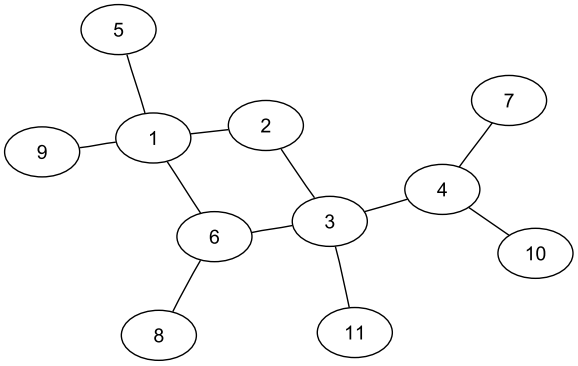

Figure 2.1

Example Distance Vector Network Topology

2.2 Dynamic Destination-Sequenced Distance-Vector Routing

Dynamic Destination-Sequence Distance-Vector routing (DSDV) is a very interesting protocol. The protocol solves the count to inf nity problem that is experienced in the basic distance vector protocol. This is accomplished by the introduction of a sequence number associated with the source of the information and including the next hop with each routing update. The sequence number retains an even value if the route is valid, but changes to an odd value if the route destination becomes unreachable. The additional overhead ensures loop free operation without the count to inf nity problem, assuming that the system has reached a steady state [16].

The addition of a sequence number and the next hop is not the only change to the basic DV protocol introduced in DSDV. Some events also trigger a reactive response which causes a node to send an intermediate update which only contains changes since the last periodic update [16].

DSDV also includes additional data within the routing table that each node maintains to reduce network overhead. The routing table consists of the following values: destination, next hop, metric, install time, stable data[16]. The metric is the hop count to a given destination. Install time is the time at which the route was added to the table, while stable data is an aggregation of data that represents the amount of time before receiving the first route for a particular destination and the time until the best route will be received[16]. This data is used to determine how long to wait before sending a route update to prevent flooding the network with excess traffic after each and every change[16].

2.3 Secure Routing Research

Research in secure routing is mostly contained within three categories: malicious activity detection and mitigation, cryptographic route authentication, and the use of tamper resistant devices to ensure adherence to the rules by participants [8, 23].

2.3.1 Without Trusted Computing Modules

Kimaya Sanzgiri et al. [18] propose the Authenticated Routing for Ad hoc Networks (ARAN) protocol for securing ad hoc networks. The proposed protocol requires that all nodes wishing to participate receive a certificate from an online trusted authority and use

asymmetric cryptography for route authentication [18]. This requirement of a central authority is unrealistic in many types of ad hoc networks.

Secure Ad Hoc on Demand Distance Vector Routing (SAODV) [23] is an extension of AODV. Like ARAN, SAODV makes use of asymmetric cryptography to authenticate routing information. However, the use of asymmetric cryptography is taxing [23]. This result, though not entirely unexpected, is useful in realizing that asymmetric cryptography is too expensive in terms of computational requirements.

Yih-Chun Hu et al. [7] developed the Secure Eff cient Distance Vector Routing (SEAD) protocol which addresses Denial-of-Service (DoS) attacks. SEAD provides authentication for sequence numbers and the distance value through the use of hashing functions. This approach reduces the computational overhead of authentication as compared with asymmetric cryptography.

Ariadne[9], developed before SEAD, is yet another attempt to secure routing protocols. Another protocol, TESLA, is used to handle broadcast authentication for Ariadne without the use of asymmetric cryptography[9]. It uses commitment keys and hashing algorithms for authentication. This is done by loose time synchronization between the nodes and the publishing of keys for authenticating prior messages[9]. One attack against Ariadne allows a malicious node to create an authenticated route that does not exist [6]. The attack is possible due to the malicious users being able to collect locality information based on route requests through the node or nodes they control. Secure systems should not allow malicious users to propagate f ctitious routes.

Secure Dynamic Source Routing (SDSR) [13], as the name implies, is another DSR based secure routing protocol. SDSR makes use of asymmetric cryptography. One primary advantage to SDSR is that the nodes do not require the existence of a trusted authority.

2.3.2 With Trusted Modules

While the use of secure coprocessors for routing is not a new new idea, current work does not address the division of labor between the network agent and user agent. Sean Smith of Los Alamos National Laboratory suggests the use of secure coprocessors, "in every step of a communications and computation path [19]." Joo-Han Song et al. [20] propose the use of secure coprocessors specif cally for secure routing. The proposal involves integrating medium access control (MAC) into the secure coprocessor. The MAC is the portion of a networking device that mediates access to the physical medium where data is transferred. The addition of the MAC drivers within the trusted boundary may be feasible, but due to the wide variety of MAC layer protocols, and for the purpose of rendering this research to have a broader applicability, in this research we do not assume that the MAC layer is under the control of the network agent. Furthermore, extending the capability of the tamper resistant module will also increase its cost. For such a device to be prolif c, the device must be inexpensive especially in the case of wireless sensor devices.

In addition to introducing complexity, there are many problems that arise from including the MAC layer within the bounds of the tamper resistant device. One of the primary reasons for not including the MAC layer is that technology changes rapidly. A secure device, once built, cannot be disassembled and upgraded to the new technology. A pos-

sibly more important concept is that if an entity wishes to build a secure coprocessor it becomes application specif c. For example, it must contain a wireless communicator or wired connector. If one wishes to use the same device for many types of networks, they must now include all of the MAC layer components for each type of network the device is to interact with. With all of the additional components the device is now more expensive and potentially less secure.

Michael Jarrett and Paul Ward propose the Trusted Computing Ad-hoc On-Demand Distance Vector (TCAODV) [10] routing protocol. This protocol relies on the Trusted Platform Module and asymmetric cryptography. TCAODV attempts to verify the successful transmission and receipt of messages from the physical hardware, in fact, it relies on this ability to provide assurances of security [10].

Levente Buttyán and Jean-Pierre Hubaux introduce the idea of nuglets [5] to ensure the participation of parties in routing. A nuglet is a counter used as a form of currency for sending data. By forwarding data and routing information a node earns nuglets; a node then uses a nuglet by sending its own data. The counter is stored within the secure coprocessor. The idea is quite interesting since it provides nodes with the option of participating or not depending on the users interests. This approach only addresses the wants and needs of the user and the needs of the whole are not discussed.

CHAPTER 3

DUAL-AGENT APPROACH

Before introducing the approach it is necessary to describe two data structures that will be used in the approach: Bloom f lters and next secure f eld.

3.1 Data Structures

3.1.1 Bloom f lters

Bloom f lters were f rst introduced by Burton H. Bloom [3]. These f lters provide a randomized f xed-space data structure for inserting and querying the existence of an item within a set. It is important to note some key facts about Bloom f lters. First a Bloom f lter will never return a false negative, but there is a potential for false positives. The probability for false positives can be determined in advance if the maximum number of items the set will contain is known [4]. Once an item has been added to the f lter it cannot be removed. A Bloom f lter is def ned by three variables,

- m - the number of bits used to represent the f lter,
- k - the number of hash functions used to insert and query items, and
- n - the number of items the f lter holds.

Initialization of a Bloom f lter is done by zeroing all m bits of the Bloom f lter. Initialization can occur at any time, clearing all items from the f lter.

Items are inserted into a Bloom f lter by hashing the value of an item by each of the k hash functions and setting a corresponding bit within the Bloom f lter for each hash function. The hash functions should be independent and map an item to one bit per hash function within the f lter[4]. Once a bit is set it will not be cleared until the Bloom f lter is initialized again. Figure 3.1 demonstrates the insertion of messages m_1 and m_2 into an empty Bloom f lter with $k = 2$.

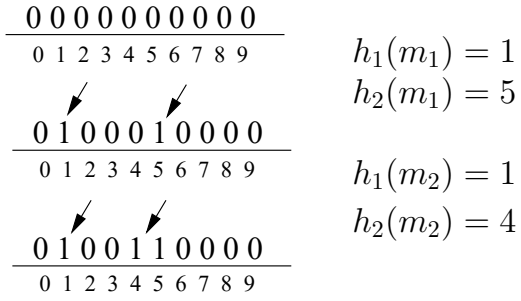

Figure 3.1

Inserting m_1 and m_2 into Empty Bloom f lter

An item is queried by hashing the value by each of the k hash functions and checking if the corresponding bits are set in the f lter [3]. If any of the bits are 0 then the item is not contained within the set. If all the bits corresponding to each hash function are set then either the item is in the Bloom f lter or a collision has occurred. A collision occurs when a value being inserted hashes to one or more values already contained in the f lter or when an item queried hashes to locations previously set by one or more messages.

3.1.2 Domain Name System Security Extensions

Domain Name System Security Extensions (DNSSEC) [1, 2] is a massive undertaking to secure the Domain Name System (DNS) for the Internet. The interest with regard to this work is specif c to the next secure (NSEC) [2] chain combined with public key cryptography as described in IETF RFC 4034[1]. Public key cryptography is vital in DNSSEC because it allows any and all users to verify the authenticity of the records. The NSEC chain is the key element in providing a method for authenticated denial of DNS entries. When requesting a DNS record the server must respond with one of two things: either the DNS record requested or a record for the value that would immediately precede it [2].

A simplif ed section of the NSEC chain for *test.com* is shown below in Figure 3.2. Assume a user makes a request for a DNS record for a domain name *cts.test.com* to the DNS server. The server does not currently have an entry for the requested domain name, but the server must prove this to the requester by providing the value that proceeds the request. For the example the server would return the record associated with *bts.test.com* which contains the the NSEC f eld *ets.test.com*. This informs the requester that the server does not have an entry for the requested domain name. Since each record is signed, the requester is able to authenticate the record and infer the absence of the requested domain name in the DNS server.

Each set of records has a validity time frame so that old records cannot be presented as current and accurate[2]. This data structure works well for DNS servers since the records do not change very frequently. When a record corresponding to a new domain name name

[1] A more thorough explanation of DNSSEC can be found at `http://www.dnssec.net`

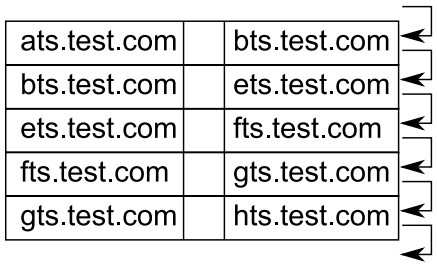

Figure 3.2

Simplif ed NSEC Chain Example

is to be added, all of the items in the chain are then updated and signed at the next time interval[2]. The NSEC chain as described in the specif cation does not allow for dynamic updates. Updates occur only when validity time of the current records is about to expire and a new set is generated[2]. In the time between these intervals the records cannot be modif ed, as this would create the potential of having more than one signed record allowing the DNS server to decide which it would like to present to requesters.

3.2 Approach

The primary motivation for the dual agent approach stems from the need to separate tasks that further individual interests from those that further the interests of the community. The user in control of a mobile device is inherently interested only in the individual needs and wants, while routing requires the cooperation of the community. This disjoint is a primary reason to investigate the division of tasks between two entities. The network agent's goal is to maintain adherence to rules which are necessary for the community. The user agent must appease the wishes of the individual. The dual-agent approach wishes to

enumerate the tasks that the community must protect and those that the individual wants to protect. Once the tasks are enumerated, the network agent's job should be reduced to the ability to *verify* that the user agent is fulf lling the requirements of the community.

Apart from low capability, the network agent is also limited by the fact that the user agent is in physical possession of the network agent. The user agent can power down the network agent at any time. Furthermore, all communication interfaces are under the control of the user agent. Thus, any interaction between two network agents has at least two untrusted user agents in the middle. These inherent restrictions on the network agent have to be considered in the protocol design.

The strategy adopted in this thesis is to divide the tasks based on the their impact on the community. The second step is to determine a set of enforceable rules to prevent or mitigate misbehavior. These rules will necessarily involve some level of redundancy to verify that tasks are occurring as they should. Such redundancies themselves will consist of extra tasks to be performed by the user and network agents. It is in our interest to reduce the extra tasks assigned to the network agent even if it results in a large increase in the user agent tasks.

3.2.1 Dependent and Independent Tasks

It will be the case that some set of basic tasks are not dependent on other tasks or events, while some may be dependent on existing information or the completion of other tasks. For example, a route update is comprised of routes received over time that have been aggregated into a routing table. Before signing a route update the network agent will

require proof that the current advertisement is consistent with the routing updates that had been previously received from other nodes. On the other hand, for periodic *alive messages* (to aff rm a client is still available), no proof is needed. However such messages could be controlled by rules that govern the rates at which they can be sent (for example once a minute). While the network agent is unable to control the medium it can refuse to provide an authenticated *alive message* to the user agent. Packets sent without authentication by the network agents will simply be dropped by other nodes (or the network agents of other nodes).

3.2.2 Attacks and Countermeasures

The model of the dual-agent approach introduces several types of attacks that exploit the weaknesses of the network agents in regards to their ability to verify the functions of the user agent. Such operations are performed on behalf of the network agent and the community. Since the network agent should have extremely low complexity, the network agent can store little to no history of network activity. This limitation opens the network agent to attacks such as: fabrication, selective dropping, omission, and replay.

Fabrication in the context of this thesis will refer to the attempt of an adversary to introduce the existence of a route that does not exist. The act of selective dropping will refer to a user agent not providing packets received for validation and inclusion by the network agent. Omission will refer to an act by a user agent where route information that had been previously validated is omitted by the user agent when network agent is

requested to sign out-going information. Finally, replay will refer to representation of previously revoked routing information to the network agent by the user agent.

Fabrication can be mitigated by authentication. For the purpose of this research the hop to hop authentication is used to counter data fabrication. Each network agent will be pre-seeded with a key ring that will be used for authenticating other network agents. This research will not discuss the issues associated with pre-key distribution as it is outside the scope of this research.

The act of selective dropping by the user agent is slightly more diff cult problem. Since the user agent has complete control of the communication medium the user could drop data indiscriminately or selectively. This thesis chooses to counter the problem of selective dropping through hop-to-hop encryption. The act of indiscriminately dropping packets is impossible to counter in this model because the user agent has control of the access medium. Since encryption hides the contents of the message, and the user agent is unable to decrypt the data. The user agent must provide the information to the network agent before being able to determine the contents.

The act of omission of a route by the user agent will be countered using the next route f eld similar to the NSEC f eld in DNSSEC. It is important to note that unlike DNSSEC, the network agent is the only party verifying the consistency of the data; therefore, public key cryptography is unnecessary. Symmetric cryptography based on a key hidden within the conf nes of the network agent's trusted boundary suff ces. A Hashed Message Authentication Code (HMAC) could be used where the hash function could simply employ the hardware block cipher.

Unfortunately, the NSEC field alone is not capable handling the dynamic nature of a typical network. The network changes frequently and a data structure capable of storing revoked messages is needed. Constant-time insertion and query are desirable traits in addition to having minimal space requirements. It is highly likely that different routing protocols will require custom data structures to verify that revoked messages are not passed as a valid routes. The use of bloom filters are utilized for the dual-agent distance-vector protocol. Cuckoo hashing[14] and hash compaction[21] are other methods which produce similar results. Cuckoo hashing requires a rebuild of the table when collision loop occurs[14], the requirement of rebuilding the table is unacceptable for the dual-agent approach. Hash compaction could potentially reduce the size of the structure needed but it does not have constant time insertion; future research could consider the implications of this constraint. There are still many other such storage schemes but bloom filters are relatively simple to implement and as such were considered the best choice.

CHAPTER 4

DUAL-AGENT DISTANCE-VECTOR ROUTING

In this thesis, the broad dual agent strategies discussed in the previous chapter are applied for securing distance-vector based routing protocols. The distance vector protocol borrows many elements from from DSDV. The dual-agent protocol takes advantage of using even sequence numbers to represent reachable route destinations and odd sequence numbers to represent unreachable route destinations. The metric value of an invalid route is not set to inf nity. The dual-agent DSDV protocol does not handle intermediate updates and only relies on periodic updates making it strictly a proactive protocol (and not a hybrid one like DSDV). Other intricacies from DSDV are omitted for the sake of simplicity.

The dual-agent protocol has three main components: a routing table structure, a routing update structure, and a Bloom f lter revocation structure. The structure of each of these components is outlined in the sections below.

4.1 Routing Table Structure

One of the key data structures in any routing protocol is the routing table maintained by each client. In the case of the dual-agent protocol the routing table is maintained by the user agent. Routes representing neighbors are not included within the table. The routing

table is initialized by the network agent at the user agent's request. The f eld layout of the structure is presented in Figure 4.1.

destination	next hop	metric	sequence number	end of life	table item HMAC	next route

Figure 4.1

Route Record Fields

The canonical ordering used for the routing records in the routing table are based on the destination f eld. The next route f eld points to the destination of the next route record in the table. The last element in the table will point to the f rst element in the table. An example of the route table is shown in Figure 4.2. With this layout, a user agent can prove to the network agent that it does or does not have a route to a specif c client in the network.

4.2 Routing Update Structure

At each periodic update a routing update is generated by each node in the network and broadcasted to neighbors. The structure of a route update is presented in Figure 4.3. The structure contains the following information:

- creator–Id of the network agent responsible for this update (must be known for authentication)
- update count–The number of items in the list of updates
- list of updates–A list of updates ordered by their destination f eld
- header HMAC–Authentication appended to verify the integrity of the header

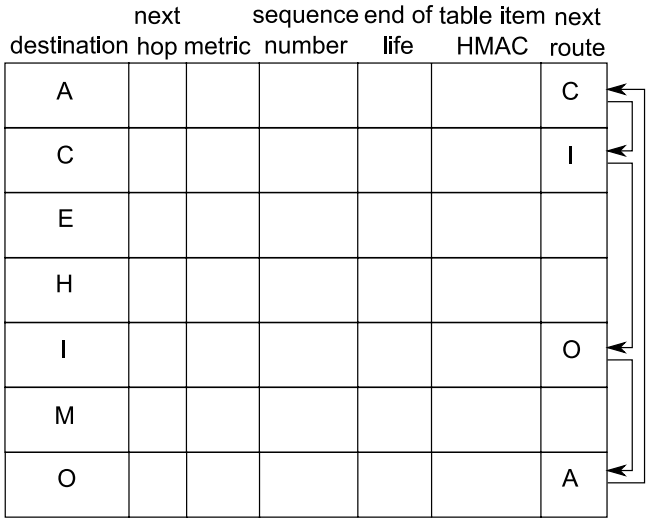

Figure 4.2

Route Table containing Route Records

Each update entry contains the following fields: destination, next hop, metric, sequence number, time of creation, and item HMAC.

- destination–Destination associated with route
- next hop–Next hop taken as part of the route
- metric–Hop count associated with reaching the destination
- sequence number–Field that is updated by the destination during a broadcast
- time of creation–The time associated with when the original route was broadcast
- item HMAC–Authentication appended to verify the integrity of the route

creator	destination	update count	header HMAC

destination	next hop	metric	sequence number	time of creation	item HMAC
destination	next hop	metric	sequence number	time of creation	item HMAC
destination	next hop	metric	sequence number	time of creation	item HMAC
destination	next hop	metric	sequence number	time of creation	item HMAC

Figure 4.3

Route Update Field Structure

The time-of-creation of a record is used to compute the end-of-life of the record. However different nodes will compute different end-of-life's depending on the number of hops that separate the node from the destination (which specif ed the time-of-creation). More specif cally, the end-of-life for a record indicating time-of-creation T_c, for a node that is h hops away is

$$T_e = T_c + xT_i + hyT_i, \text{ where} \tag{4.1}$$

- T_i the length of the time interval between updates
- $x \geq 1$, and
- $y < 1$

The header HMAC and item HMAC is keyed with the key associated with the neighbor from with the update was received from or to be sent to.

4.3 Protocol Operation

Operation begins with a request from the user agent to initialize the network agent and the routing table. The network agent responds to this request by returning a route record that represent the client itself. The record for client 4 would look like $(4|4|0|0|0|HMAC|4)$ where $HMAC$ is the HMAC of all of the f elds in the route table item and a private key only the network agent has.

Eventually, route updates received by the user agent will need to be merged with the existing routing table. The routes contained within the updates will either new routes or updates to existing routes.

- new–The route table does not current contain a route to the destination.
- updates–A route that the route table does currently contain a route to.

Before adding a new record to the route table, the user agent must provide the a route record with destination less (in the canonical ordering strategy used) than the destination indicated in the new route record to be added. The record supplied as proof should indicate, in the next route f eld, a destination greater than the one in the new record.

Before or updating a route record in the routing table, the user agent must provide the network agent with the route record that the network agent is attempting to update based on the route update.

Any addition of records or deletion of records will result in invalidated records. For example, consider a scenario where a node currently has valid records for destinations $4, 6, 8, \ldots$. If a new record (say for destination 5) is added the record for destination 4 should be modif ed to point to 5. The old record (which pointed to 6) should be expunged.

Similarly when a record (say 6) is deleted, the record for destination 4 should be modif ed to point to 8 as the next route record.

It is necessary for the network agent to keep track of all expunged records. If this is not performed, the user agent, who has a record for 5, can *replay* the old record (indicating 6 is the record that follows 4) to deny the fact that it has a route to 5. It is obviously desirable to maintain the list of invalidated records for as small a duration as possible. This is the primary reason for assigning a specif c life-time to every record stored.

If the life time is f xed, and kept small, many records may expire even before they reach nodes many hops away. This is the reason that the hop-count is taken into consideration for computing the end-of-life.

The enforcement of life times to every record calls for some extent of time synchronization. Furthermore some records may have to be deleted due to expiry of life-times. This can happen even before they are superseded by newer routing records. Thus, prior to expiry of any route record stored by the user agent, the user agent has to inform the network agent. The network agent has to create a new record which replaces the old previous record (which points to an invalidated next route). The old previous record (which is still valid as the end-of-life may be far into the future) needs to be expunged. If this task is not performed by the user agent, the client's ability to participate in the network will be diminished as it will be unable add new records, and ultimately to send routing updates.

Both the user and network agents need to keep track of invalidated records, which however indicate a end-of-life in the future. Bloom f lters were chosen as the data structures for this purpose, due to their f xed-time insertion and query complexity. Additionally, the

reduced storage requirements of Bloom f lters make them an good candidate for storing invalidated records. Further, the hash functions for the Bloom f lter can also be eff ciently implemented using the block cipher.

More generally, the user and network agents can maintain a set of many Bloom f lters, each of which represents some range of time in the present or future. The exact number and range of such f lters were determined experimentally. Recall that when a route record is removed, updated, or added, one or more revoked messages are produced. Each of the messages that have an end of life in the future are placed into a specif c Bloom f lter whose time-range includes the records end-of-life. Later, when a network agent is presented with a message it can to query the specif c Bloom f lter based on the end of life to determine if the message is a replay attempt. The Bloom f lters covering the range T_1 to T_2 should be maintained till time T_2. After T_2 the Bloom f lter can be discarded. In summary, only items that have been revoked whose end of life has not passed are placed into a Bloom f lter.

The user agent should be aware if a collision occurs when attempting to add a new message to the Bloom f lter. If a collision occurs the user agent should provide all of the messages that produced the collision to the network agent. When the route and associated data has been verif ed by data from the prior messages, the network agent accepts the new message as valid.

After receiving a route that is more recent (higher sequence number), or just as recent while having a lower hop count (lower metric), the previous route is still valid until the end

of life has passed. To counteract the ability to have two valid routes, the old route must now be added to the Bloom f lter corresponding to the end of life of that route

4.3.1 Example Routing Table Changes

To add a route, the network agent expects the user agent to provide proof in the form of the route immediately preceding it canonically and the route to be added. To update an existing route, the network agent expects the user agent to provide the current route in the routing table and the new route. This new route could have a better metric or a higher sequence number.

For the following example, the clients *A*, *B*, *C*, *D*, and *E* are connected as shown in Figure 4.4.

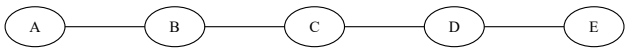

Figure 4.4

Example Network

The broadcast order will be alphabetical and each client will broadcast only once per interval. Just after initialization each client will only have a self route. Client *A*, for example, has only the route $(A|A|0|0|0|HMAC|A)$, and client *B* has only $(B|B|0|0|0|HMAC|B)$. Client *A* broadcasts f rst which causes changes to the state of *B*'s routing table. The changes do not require any bloom f lters to be tested because the route learned was a neighbor. No route representing a neighbor is used to indicate a next route. This is re-

ferred to as the *neighbor route rule*. The "-" is used to indicate that a neighbor will not have a next route and no other next route points to it.

- $(A|A|1|0|3|HMAC|-)$
- $(B|B|0|0|0|HMAC|B)$

Still within the first time interval, the route table of *C* is $(C|C|0|0|0|HMAC|C)$ before *B* broadcasts. *C* receives two routes from *B*, the route to *A* and the route to *B*. The routing table below represents the changes after processing the first update from *B*. No items need to be added to a bloom filter because the only route altered is the self route. If the route modified is the self route nothing must be added to any bloom filter. This is referred to as the *self route rule*. The processing of the route to *B* falls under the neighbor route rule and is not shown.

- $(A|B|2|0|4|HMAC|C)$
- $(C|C|0|0|0|HMAC|A)$

While still in the first interval, the route table for client *D* is $(D|D|0|0|0|HMAC|D)$ before receiving an update from *C*. When processing route *A* to the table the self route rule applies and no additional steps are performed.

- $(A|C|3|0|5|HMAC|D)$
- $(D|D|0|0|0|HMAC|A)$

When adding route *B* to the table no current rules apply. Route *A* is not a neighbor and is not the self route. Any changes to this route are not protected by the network agent unless it is added to the next route chain. The old route *A*, $(A|C|3|0|5|HMAC|D)$, must be provided to the network agent as proof. If the route is shown to be valid the route

must be added to a bloom filter and the new route as shown below would replace it. This route must first be tested for existence in the bloom filter containing routes that expire in interval 3. Assume that a bloom filter represents a full time interval and the route expires in interval 3, the bloom filter with duration interval 3 (inclusive) to interval 4(exclusive). This is referred to as the distant route rule. If the addition caused no collision in the filter when adding the route to the bloom filter then nothing else need be done. However, if the addition did cause a collision it would need to be hashed to a 16bit value and stored separate from the bloom filter.

- $(A|C|3|0|5|HMAC|B)$
- $(B|C|2|0|4|HMAC|D)$
- $(D|D|0|0|0|HMAC|A)$

4.3.2 Example Bloom Filter

The user agent should be aware if any route provided as proof caused a false positive within the bloom filter queried. If a false positive occurs the user agent must provide one or more routes that are invalid that caused the collision to occur. These routes are hashed and verified to be the cause of the collision. The invalid routes provided must have and end of life corresponding to the duration of the bloom filter queried where the collision occurred. This is important to note because when a collision occurs it requires more processing for the network agent to verify. Further, a new value to be added to the bloom filter and causes a collision that item is stored in a 16bit hash and not in the bloom filter. In this way the list can be consulted to ensure that the user agent is not using a collision to undermine the security of the system.

Any time a bloom f lter must be consulted for verif cation the route provided as proof must be queried in the bloom f lter and any 16bit hashes stored outside the bloom f lter. If it is shown as a hit in the bloom f lter, the user agent must provide the messages that caused the collision, each of these messages are queried in the bloom f lter to verify that they do in fact cause the collision. If any of these messages hash to the 16bit hashes then the message provided is invalid and cannot be used as proof. This is based on the assumption that 16bits is enough to not have another collision within the the collection of 16bit hashes.

For some bloom f lter, a collision has occurred on route $(B|C|2|4|6|HMAC|D)$ the user agent now provides (as example) $(A|C|3|2|6|HMAC|B)$ and $(E|D|2|4|6|HMAC|A)$ which has to the same values as $(B|C|2|4|6|HMAC|D)$. The network agent hashes the two messages to verify the collision and also verif es they are in fact not any of the messages in the collection of 16bit hashes associated with the bloom f lter.

CHAPTER 5

EXPERIMENT DESIGN

The basis for revocation in the dual-agent distance vector protocol is the assumption that a limited number of insertions and queries will be made during the time intervals. Having a limited number of transactions reduces the computational overhead on the network agent. The Bloom filters will need to be maintained until such time the elements within one have all expired. This assumption must be verified through experimentation.

There are currently multiple factors that could potentially effect the necessary size and number of Bloom filters required. Network size, client status changes, and number of neighbors. A baseline must be established to test some of these factors. The baseline for the experiment will be based on a network of one hundred clients with the clients having an average of five neighbors. Further, each client will have a 10% probability of changing state. Changing state will cause a node that is awake to sleep and a sleeping node to awake. For the purpose of the experiment physical movement of the clients will not be modeled. Time will be divided into two types of divisions. The first will be referred to as an *interval* which represents the amount of time for all the clients in the network to broadcast their routing information once. The second will be called a *tick* which represents some unit of time that a single node uses to communicate its route updates to all of its neighbors. The baseline experiment will monitor the one hundred clients for one hundred intervals.

During this time each revocation will be recorded and at the beginning of each interval each client's routing table will also be recorded, for later analysis.

Two different processes will evaluate the data recorded. The first will parse the tables of all the clients at each interval and compute the following metrics about the routes for each node:

- Invalid entry (I_E)–Client has a route entry but the destination is physically disconnected from the source
- Invalid route (I_R)–Client has a route entry but the route is unable to deliver to the destination
- Valid entry (V_E)–Client has a route entry and the route is able to deliver to the destination or the destination is unreachable and the client is aware of this
- Path Exists (P_E)–Client does not have a route entry but a physical path exists
- No Path (N_P)–Client does not have a route entry and no physical path exists

From the results of these metrics the following formula will be calculated to determine the efficiency (E), route accuracy, of the protocol:

$$E = \frac{V_E}{V_E + I_E + I_R + P_E} \tag{5.1}$$

The minimum lifetime value (xT_i from Equation 4.1) is an initial value for the end of life. By adjusting the minimum lifetime, the number of Bloom filters required at any given time is altered; additionally, if the value is set too low the network is unable to reach a stable state where routes exist for all nodes. Figure 5.2 shows the results of minimum lifetime on the efficiency of the network. As the minimum lifetime is increased more Bloom filters or larger Bloom filters are required to maintain the system.

The average efficiency metric will be computed for three network sizes to determine the effect of adjusting the minimum lifetime of the efficiency of the network.

While trying to achieve a high level of routing efficiency, we should simultaneously strive to reduce the space complexity of the Bloom filters. The optimal choice of the number of Bloom filters and the parameters of each filter will depend on the number of invalidated messages that need to be stored at any time.

In our simulations all invalidated records, indicating their end-of-life and the time at which revocation occurred, were stored to facilitate off-line analysis of optimal parameters. Off line analysis involves fixing the bloom filter duration of each filter (for example 1 time interval or $\frac{1}{2}$ time interval or 4 time intervals) and computing the probability distribution of the number of items added to each Bloom filter.

Equation 5.2[4] is the asymptotic approximation of the probability that a collision has occurred in a bloom filter, where

- k is the number of hash functions,
- m is the bit size of the bloom filter, and
- n is the number of elements in the filter.

$$f = \left(1 - \mathrm{e}^{-\mathrm{km/n}}\right)^k \tag{5.2}$$

Let $p_i(n)$ be the probability that n elements were added to a Bloom filter of length m, utilizing k hash functions. The expected number of collisions that can occur (during the time a particular Bloom filter is actively maintained) is now:

$$C(k,m) = \sum_{n=1}^{\infty} p_i(n) n \left(1 - \mathrm{e}^{-\mathrm{km/n}}\right)^k \tag{5.3}$$

In practice, when a collision occurs, the network agent cannot add the record to the Bloom filter. The records should be maintained separately (in addition to the Bloom filter). For

example, a succinct hash of the record could be stored by the network agent. It is probably reasonable to attempt to choose the parameters k and m such that $C(k, m)$ is around 1. By choosing the value at 1 we assume that on average only one has will need to be stored in addition to the bloom f lter. The choice of a 16bit hash seems reasonable to assume that no collisions will occur between such a small set. Equation 5.4 shows the space complexity ($s(f, k, m)$) for one network agent; let f be the number of bloom f lters a network agent must maintain.

$$s(f, k, m) = f\,(m + C(k, m) \times 16) \tag{5.4}$$

5.1 Simulation Results

5.1.1 Eff ciency

For the baseline several values were f xed and ten runs were performed to determine the best possible results that could be expected. The probability a client is toggled alive or inactive for an interval is set a 10%. Each simulation consisted of 100 clients randomly distributed on a unit square who's radius of communication was adjusted until the clients had an average of 5 neighbors. The simulation lasts for 100 time intervals. The minimum lifetime, a value that directly effects the end of life, was set to a value that placed the end of life beyond the scope of the simulation. This allows for a baseline or maximal measurement for the eff ciency of the protocol to be determined. The average of the runs is displayed in Figure 5.1. The graph plots the eff ciency versus the time interval the data was collected. The average maximum eff ciency rating for a 100 network with 10% client f uctuation is approximately 90%.

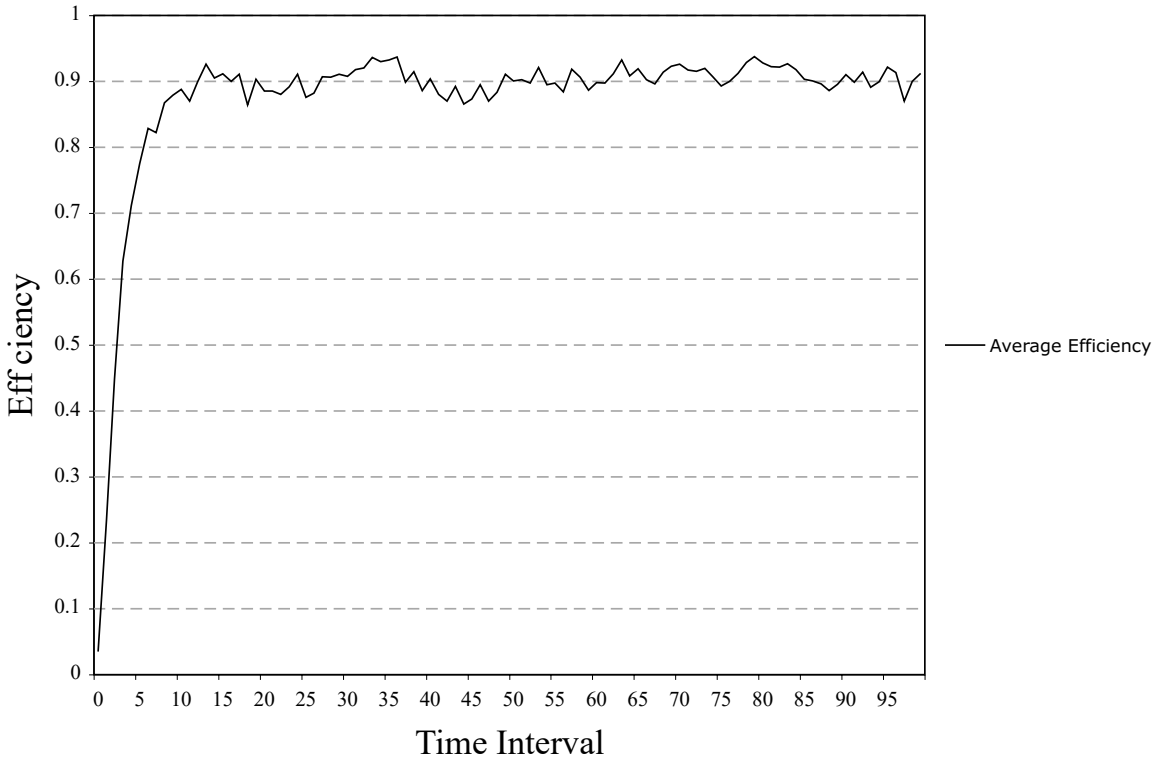

Figure 5.1

Average Network Eff ciency with Extreme Minimal Lifetime

The impact of network size will also be important to determine for the use of the dual-agent protocol. The assumption was that the choice of minimum lifetime would be similar despite network size. This assumption was tested for a networks of size 100, 200, and 400. The values chosen are based on the assumption that 400 or fewer nodes is a reasonable number for a proactive protocol. The minimum lifetime still appears to remain an accurate value in determining the eff ciency of the network regardless of network size.

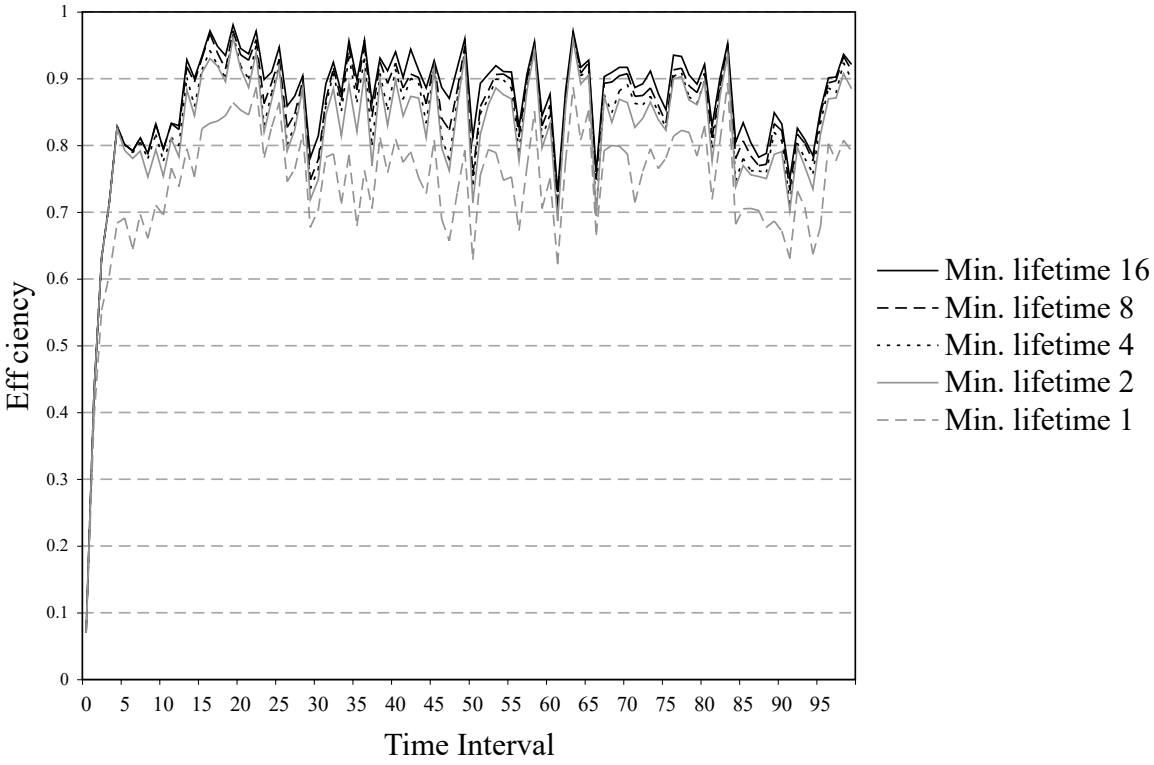

Figure 5.2

Average Eff ciency of Size 100 Network

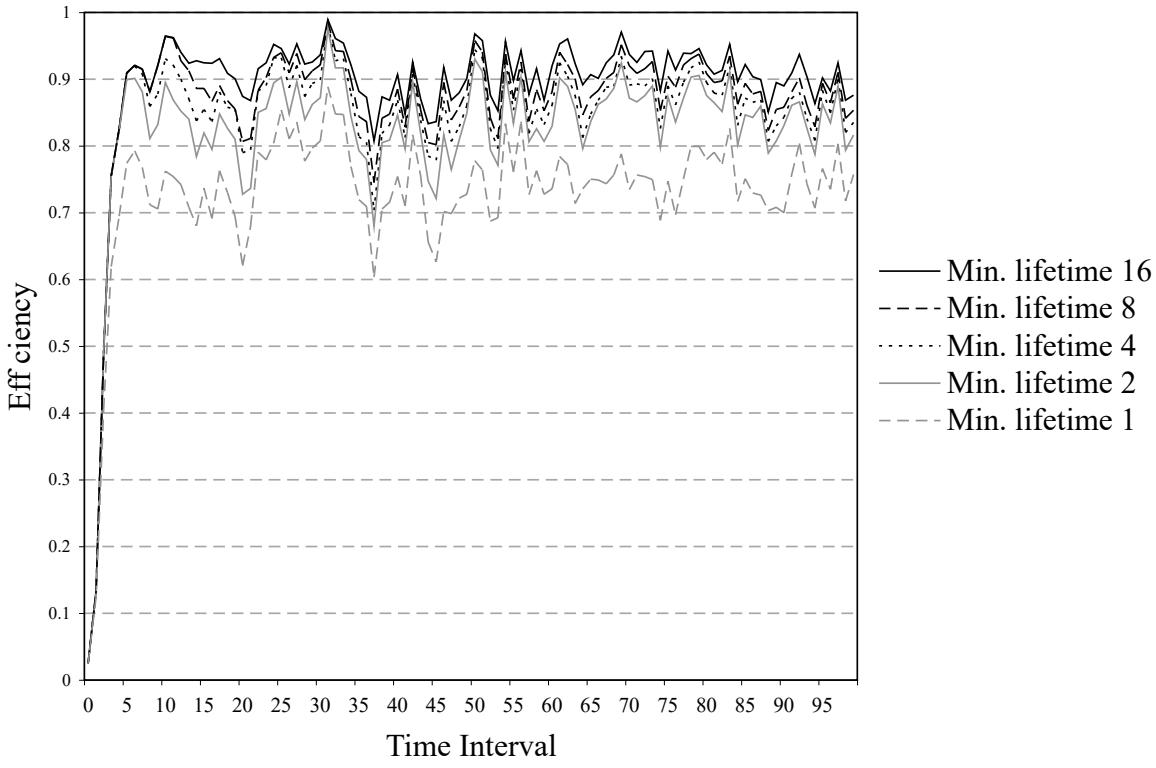

Figure 5.3

Average Eff ciency of Size 200 Network

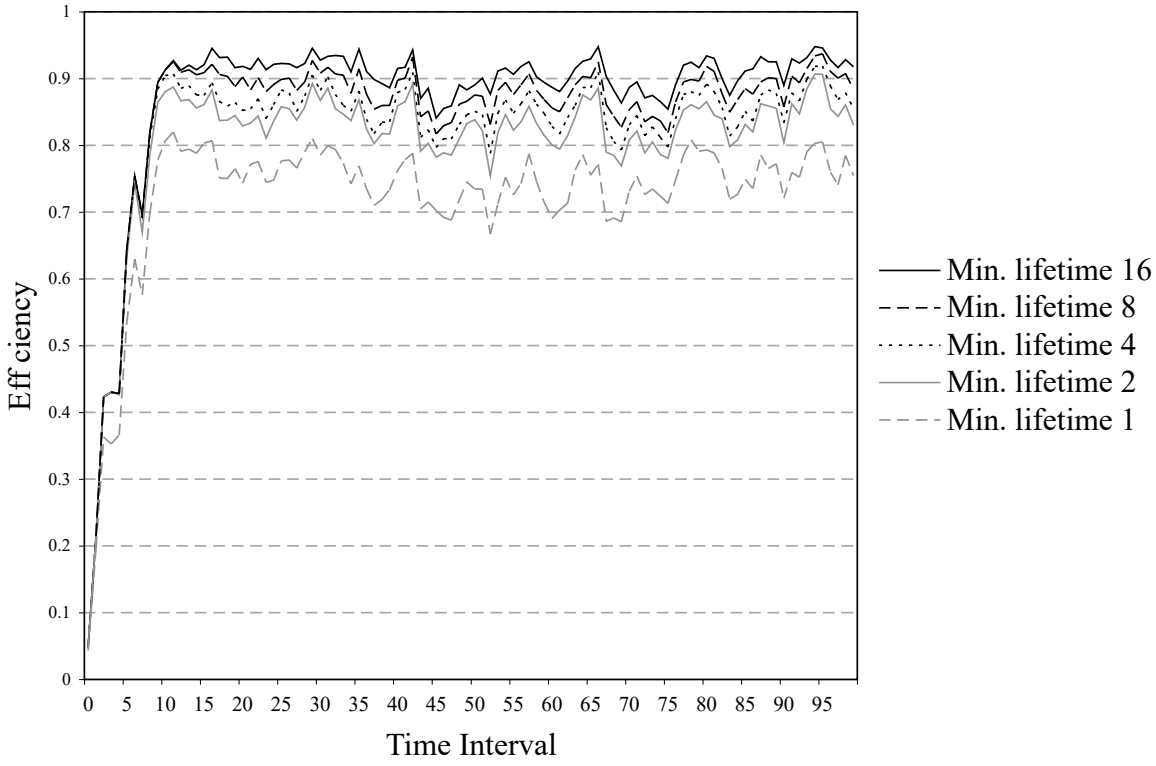

Figure 5.4

Average Eff ciency of Size 400 Network

5.1.2 Bloom Space Complexity

The Δ (Bloom f lter duration) was tested at the values of 10%, 25%, 50%, 100% and 200% of the time interval. The values of the Bloom f lter were optimized for expected number of collisions of 1 and 0.1. Elements that collide would need to be stored outside the bloom f lter; the experiment assumes that the storage of each element is a 16 bit hash. A higher number of collisions, such as 1, the reduction in storage requires the network agent to perform more calculations both when inserting additional elements and when querying elements. Therefore, given an expected number of collisions of 1 each Bloom

f lter requires 16 bits of extra storage and 1 in 10 require 16 bits given an expected number of collisions of 0.1 (requiring approximately 1.6 bits per Bloom f lter). Those results are found in Figure 5.5 and Figure 5.6, respecfully.

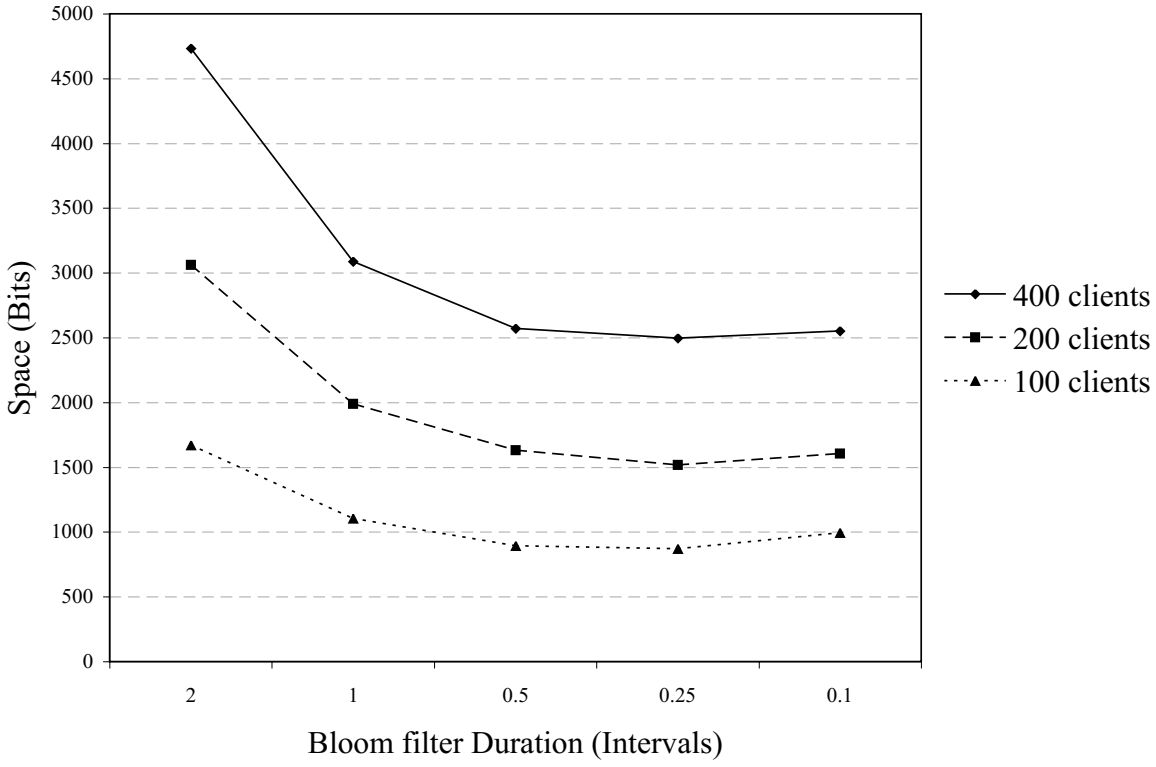

Figure 5.5

Expected Number of Collisions 1

Table 5.1

Bloom Filter Parameters with Expected Number of Collisions 1

Network Size	Δ	m	k
100	0.5	318	5
200	0.5	588	5
400	0.5	930	5

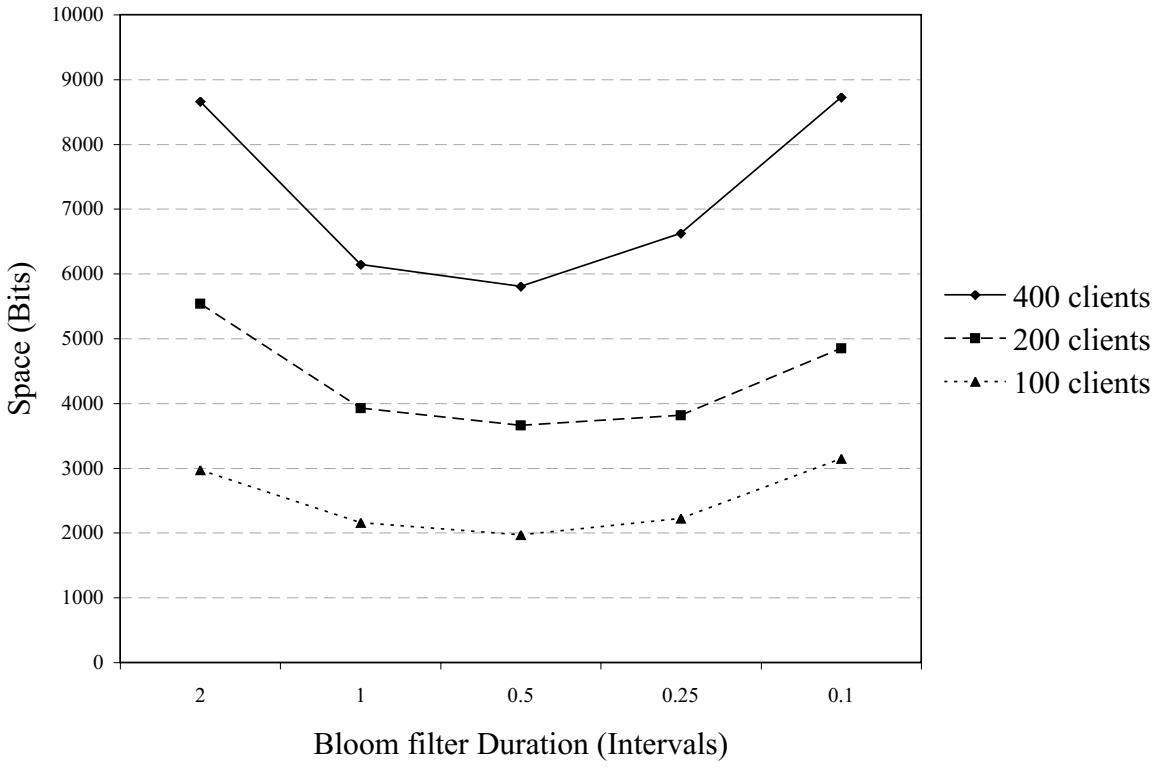

Figure 5.6

Expected Number of Collisions 0.1

Table 5.2

Bloom Filter Parameters with Expected Number of Collisions 0.1

Network Size	Δ	m	k
100	0.25	63	2
200	0.25	123	3
400	0.25	209	3

5.2 Analysis

Choosing a minimum lifetime of 2 intervals provides a high level of efficiency while reducing the number of filters need. The growth of the network has a slightly less then linear growth with respect to the average space required per node. As shown in Figure 5.7, when expecting an average of 1 collisions per Bloom filter the appropriate Δ is 0.25 for all network sizes; when selecting a lower level of collisions (0.1 per Bloom filter) $\Delta = 0.5$ provides the most efficient use of space for all network sizes tested.

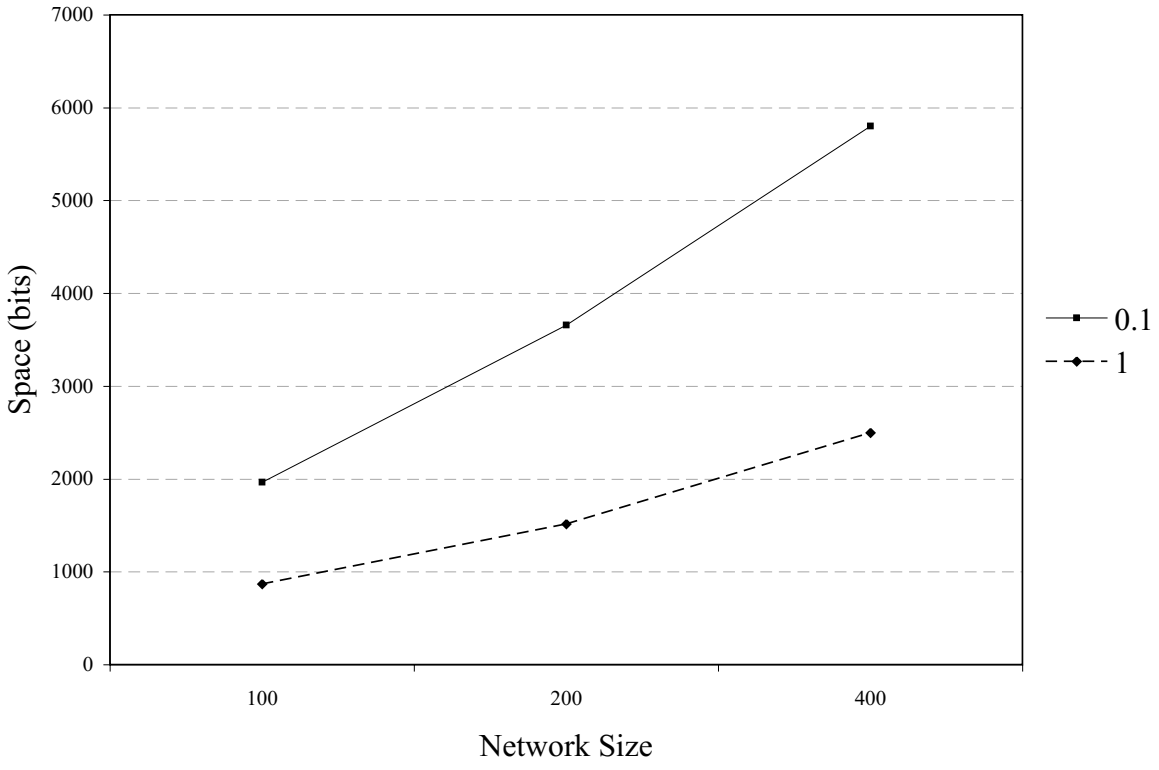

Figure 5.7

Network Size versus Space Complexity

CHAPTER 6

CONCLUSION

The results show that a high eff ciency can be achieved for a dual-agent distance-vector protocol, while keeping the memory requirements very low. The approach taken advantageously reuses a single symmetric cipher for both encryption and hashing operations. Bloom f lters reduce the space requirements of revocation lists to a manageable size even for networks as large as 400 nodes. The bloom f lters for the largest network only required approximately 2.5kB of storage space for an average of 1 collision per f lter. For a factor of 10 reduction in the number of collisions the device would still only require about 6kB of storage. This results in a slightly less then linear growth of space with respect to the network size.

This thesis identif ed four primary types of attacks that exploit the inherent limitations of the network agent: fabrication, selective dropping, omission, and replay. Further, the research has presented low complexity strategies to address each attack. These strategies were applied to a distance-vector protocol, and the resulting protocol was extensively studied through simulation to compare trade-offs between routing eff ciency and network agent complexity.

The research presented opens a plethora of new approaches to securing ad hoc routing protocols. While this thesis only considers the dual-agent approach as applied to a single

protocol; other routing protocols, such as DSR or AODV, could be considered for their applicability to the dual-agent approach. As specif c requirements of each protocol are likely to require substantially different data structures, the choice of suitable data structures could be a challenging problem potentially leading to the development of new data structures.

The f eld of secure co-processing and tamper resistant technology has opportunities to create and apply technologies to build low complexity network agents and determine best practices in their construction and interfaces.

While this thesis has restricted itself to the use of the dual-agent approach as applied to routing protocols, the approach could be applicable to f elds such as distributed computing, ubiquitous computing, and autonomic computing.

REFERENCES

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," RFC 4033 (Proposed Standard), Mar. 2005.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource Records for the DNS Security Extensions," RFC 4034 (Proposed Standard), Mar. 2005, Updated by RFC 4470.

[3] B. Bloom, "Space/Time Trade-offs in Hash coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, July 1970.

[4] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, no. 4, 2005, pp. 485–509.

[5] L. Buttyán and J. Hubaux, *Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks*, Tech. Rep. DSC/2001/001, Department of Communication Systems, Swiss Federal Institute of Technology, January 2001.

[6] L. Buttyán and I. Vajda, "Towards Provable Security for Ad Hoc Routing Protocols," *Proceedings: 2nd ACM workshop on Security of ad hoc and sensor networks*, Washington D.C., October 2004, ACM SIGSAC.

[7] Y. Hu, D. Johnson, and A. Perrig, "SEAD: Secure Eff cient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," *Proceeding: 4th IEEE Workshop on Mobile Computing Systems and Applications*, 2002, pp. 3–13.

[8] Y. Hu and A. Perrig, "A Survey of Secure Wireless Ad Hoc Routing," *IEEE Security and Privacy Magazine*, vol. 2, no. 3, May-June 2004, pp. 28–39.

[9] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Proceedings of: The 8th Annual International Conference on Mobile Computing and Networking*, Atlanta, GA, September 2002, ACM SIGMOBILE.

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

[11] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," RFC 4728 (Experimental), Feb. 2007.

[12] D. Johnson, D. Maltz, and J. Broch, *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, chapter 5, Addison-Wesley, 2001, pp. 139–172.

[13] F. Kargl, A. Gei , S. Schlott, and M. Weber, "Secure Dynamic Source Routing," *Proceedings of the 38th Hawaii International Conference on System Sciences*, Hawaii, 2005.

[14] R. Pagh and F. Rodler, "Cuckoo Hashing," *Algorithms - ESA 2001*, vol. 2161, 2001, pp. 121–133.

[15] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), July 2003.

[16] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proceedings: Conference on Communications Architectures, Protocols and Applications*, London, 1994, pp. 234–244.

[17] M. Ramkumar, "Trustworthy Computing Under Resource Constraints with the DOWN Policy," to be published *IEEE Transactions on Dependable and Secure Computing*.

[18] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "A Secure routing Protocol for Ad Hoc Networks," *Proceedings: 10th IEEE International Conference on Network Protocols*, November 2002, pp. 78–87.

[19] S. Smith, "Secure Coprocessing applications and Research Issues," *Los Alamos Unclassif ed Release LA-UR-96-2805*, August 1996.

[20] J. Song, V. Wong, and V. Leung, "Poster: Secure Routing with Tamper Resistant Module for Mobile Ad Hoc Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, New York, July 2003, vol. 7, ACM Press.

[21] U. Stern and D. Dill, "Improved Probabilistic Verif cation by Hash Compaction," *Correct Hardware Design and Verif cation Methods*, vol. 987, Springer-Verlag, London, 1995, pp. 206–224.

[22] A. Tanenbaum, *Computer Networks*, Prentice Hall, Upper Saddle River, New Jersey, 1996.

[23] M. Zapata, "Secure Ad hoc On-Demand Distance Vector Routing," *Mobile Computing and Communications Review*, vol. 6, no. 3, July 2002, pp. 106–107.