Simulation of a Secure Ad Hoc Network Routing Protocol

Anne Gabrielle Bowitz

Master of Science in Communication Technology

Submission date:	June 2011
Supervisor:	Stig Frode Mjølsnes, ITEM
Co-supervisor:	Martin Gilje Jaatun, SINTEF

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Secure wireless ad hoc networks possess many properties that are highly valuable in e.g. emergency situations and military applications. By using X.509 certificates, the ad hoc routing protocol B.A.T.M.A.N. has been modified in order to support identification and authentication of mobile nodes trying to access a restricted ad hoc network.

The new protocol design needs to be evaluated by performing tests under various conditions and environments. However, this might be a challenging and expensive task to do in a real-world system, thus it is recommended to simulate the routing protocol using a network simulator.

ns-3 is an open source discrete-event network simulator for Internet systems. The goal of this thesis is to extend ns-3 to support simulation of both the original and modified B.A.T.M.A.N. protocol such that protocol design, interactions, and large-scale performance issues can be investigated and compared.

Assignment given: 24. January 2011
Supervisor: Stig Frode Mjølsnes

Abstract

New network protocols are continuously being developed, and a particularly interesting area of research is in ad hoc networks. Due to their dynamic and self-organizing nature with no infrastructure, they introduce properties that are very beneficial in e.g. emergency situations and military applications.

BATMAN is an ad hoc network routing protocol which has been modified in order to provide an authentication mechanism which only allows authorized nodes to route traffic in the network. Routing protocols pose as a critical aspect to performance in mobile wireless networks and it is important that the modifications done for security purposes does not affect the routing performance significantly.

The goal of this study was to extend the network simulator ns-3 to support both the original and modified version of the BATMAN protocol. Then the simulator was used to study and evaluate the protocols' design, interactions, and large-scale performance issues.

Based on the observations from the simulations conducted with ns-3, the modified BATMAN protocol indicates that it does not perform significantly worse than its counterpart despite the introduced security measures.

Preface

This report is written by Anne Gabrielle Bowitz and describes the work done in my master's thesis which is a part of the Master's degree Programme in Communication Technology at the Norwegian University of Science and Technology, NTNU. The was work performed under the supervision of Martin Gilje Jaatun of SINTEF ICT Norway and Stig Frode Mjølsnes from the Department of Telematics at NTNU.

I would like to thank my supervisor Martin Gilje Jaatun which has been a true driving force when working with this thesis. Thanks for the continuous feedback on the report and for having weekly meetings! It was of great help!

Trondheim, June 27, 2011

Anne Gabrielle Bowitz

Abbreviations

AES Advanced Encryption Standard

AODV Ad hoc On-Demand Distance Vector

ATM Asynchronous Transfer Mode

AM Authentication Message

B.A.T.M.A.N. Better Approach To Mobile Ad hoc Networking

CBC Cipher Block Chaining

CBR Constant Bit Rate

DARPA Defense Advanced Research Projects Agency

DSDV Destination Sequenced Distance Vector

DSR Dynamic Source Routing

EQ Echo Link Quality

HNA Host Network Announcement

IPv4 Internet Protocol version 4

IV Initial Value

MANET Mobile Ad hoc Network

MSC Message Sequence Chart

ns-2 Network Simulator 2

ns-3 Network Simulator 3

OGM Originator Message

OLSR Optimized Link State Routing

PC Proxy Certificate

PC0 Proxy Certificate 0

PC1	Proxy Certificate 1
PDR	Packet Delivery Ratio
PKI	Public-Key Infrastructure
PKIX	Public-Key Infrastructure using X.509
REAL	REal And Large
RQ	Receiving Link Quality
SP	Service Proxy
Tcl	Tool Command Language
TCP	Transmission Control Protocol
TQ	Transmit Link Quality
TTL	Time To Live
UDP	User Datagram Protocol
VINT	Virtual InterNetwork Testbed

Definitions

Ad Hoc Network A self-organizing network with no requirements to pre-existing infrastructure or centralized administration.

Ad hoc On-Demand Distance Vector (AODV) A reactive ad hoc routing protocol [PBRD03].

Advanced Encryption Standard Cipher Block Chaining (AES-CBC) The Advanced Encryption Standard (AES) encryption in Cipher Block Chaining (CBC) mode of operation.

Authentication Fields Two fields added to the Originator Message (OGM) used in the modified BATMAN routing protocol containing extracts or one-time passwords from the Authentication Key Stream.

Authentication Key Stream Key Stream generated with the AES-CBC algorithm taking in a shared symmetric key, an IV value and a nonce as input. It is used to authenticate and tie an a node to the OGMs it broadcasts to its link-local neighbors.

Authentication Message Message used by the Secure BATMAN routing protocol containing en ephemeral key, nonce value, and Initial Value (IV).

Constant Bit Rate Data traffic generated with a constant bit rate.

Destination Sequenced Distance Vector (DSDV) Proactive Mobile Ad hoc Network (MANET) routing protocol. Routing updates are broadcasted or multicasted by every node periodically (default 15 s) and when there is significant changes in the network topology. Based on the routing updates a node calculates paths to every node in the network using the Bellman-Ford algorithm [PB94].

Direct Neighbor Refers to a node in a network which is reachable with a single hop. This is also referred to as a link-local neighbor.

Discrete Event-based Simulation Simulated network where nodes trigger events, such as a packet being sent, which is stored in a queue sorted by the scheduled event execution time [WvLW09].

Emulator A system that is able to mimic function or behavior of another system.

Empty OGM An OGM used in the modified BATMAN routing protocol containing an empty Authentication field.

Friis Propagation Model Simple transmission formula describing the propagation loss in a traffic flow between nodes [Fri46].

Host Network Announcement (HNA) Message used by a BATMAN Originator to inform other nodes in the network that it can be used as a gateway to another network or host [NALWay].

Initial Authentication Phase Phase where nodes exchange and verifies Proxy Certificate 0 (PC0) before exchanging ephemeral keys and generating authentication key streams.

Interface Address IPv4 address assigned to a BATMAN node which is put in the nodes self-generated OGMs.

Mobile Ad hoc Network (MANET) Ad hoc network were nodes can be highly mobile.

NIST ATM Network simulator targeted for simulating and analyzing the behavior of Asynchronous Transfer Mode (ATM) networks [GKS95].

ns-2 Discrete event network simulator written in C++ and OTcl based on the REAL network simulator from 1989 [nsnayc].

ns-3 Modular discrete event network simulator written in C++ and Python.

One-Time Password A 16 bit extract from the Authentication Key Stream generated by a node running the modified BATMAN routing protocol.

Optimized Link State Routing (OLSR) A proactive ad hoc routing protocol based on an optimization of the classical link state routing algorithm [CJay].

Originator Synonym for a node using the BATMAN protocol generating and broadcasting own Originator Messages (OGMs).

Originator List A table maintained by every node in a BATMAN network which consists of information about every other known originator in the network [NALWay].

Originator Message (OGM) A message periodically broadcasted by a originator to inform its link-local neighbors about its presence. [NALWay]

OTcl An object oriented extension of Tcl which is a scripting language.

Packet Delivery Ratio Defined as the ratio between the amount of received packets and the amount of packets actually transmitted.

Proactive Routing Protocol Routing protocols which periodically share routing information in order to maintain their routing tables, also known as table-driven protocol. The opposite of proactive routing protocol is the reactive protocol.

Proxy Certificate A X.509 public-key certificate including a critical certificate information extension used to delegate rights and restrictions within a network [TET+ay].

Public-Key Certificate Electronic document containing most importantly a digital signature which ties a public-key to an identity.

Reactive Routing Protocol Routing protocols that only construct routing paths when they are required, also known as on-demand protocol. The opposite of reactive routing protocol is the proactive protocol.

Restricted Network Network established between nodes using the modified BAT-MAN routing protocol where one central node acts like a Service Proxy (SP).

Service proxy A central node in a restricted network running the modified BAT-MAN routing protocol with permission to sign and issue Proxy Certificates (PCs) to other nodes.

SQLite A self-contained and serverless SQL database engine reading and writing directly to ordinary disk files [SQLay].

Symmetric Key A symmetric key exchanged between nodes in a restricted network and used when generating an Authentication Value.

VINT Project Research project funded by Defense Advanced Research Projects Agency (DARPA) which provides improved simulation tools for network researchers to use in the design and deployment of new wide-area Internet protocols [BBE+99].

X.509 Public-Key Certificate Public-key certificate standard managed by the Public-Key Infrastructure using X.509 (PKIX) working group of IETF [HPFS02].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile Ad hoc Networks (MANETs) have certain characteristics and properties that separate them from traditional computer networks. Most importantly they have no requirements to pre-existing or fixed infrastructure and they need no centralized administration maintaining the network. Here it is the participating nodes' responsibility to sustain routing paths between nodes and make sure that traffic is routed efficiently and reliably from a source to a destination.

Crisis management after a major disaster, emergency situations, and rescue operations are frequently identified as application areas for MANETs [TJÅAN09]. These are situations were resources might be scarce as well as conditions being unpredictable and rapidly changing. Existing infrastructure is unreliable as it may be damaged or congested and time is restricted. MANETs can be quickly and cost-efficiently deployed, making them very valuable in situations such as these.

The nature of the situations mentioned above imply that providing information security in MANETs is important when they are to be used here. For instance, being able to restrict the access to a network would prevent valuable resources and information being available or wasted on activities not related to the operation. Access control also enables node authentication and the possibility of confidentiality of information by only allowing authorized nodes to route traffic in the network [TJÅAN09].

Routing protocols are a critical aspect to the behavior of and performance in MANETs and pose as a natural place to apply security elements in order to achieve some sort of access control. BATMAN is an ad hoc routing protocol with a simple and robust routing regime. Its limited complexity compared to many of its alternatives, such as Optimized Link State Routing (OLSR), makes it easier to specify suitable modifications for security purposes. A modified version of the BATMAN protocol has been proposed where various security measures has been combined to provide a restricted ad hoc network.

However, with limited resources it is also important that the added security mechanisms do not substantially affect the overall performance and throughput of the network. Thus the development of such new routing protocols requires testing and evaluation against well-known protocols in various environments. Several network simulators exists providing easily accessible resources to study new protocols and models and has through time been the backbone of MANET research [NCÇ+11].

ns-3 is a discrete-event network simulator which strives to become a preferred, open simulation environment for networking research and educational use. It was developed with the focus on improving the core architecture, software integration, and simulation models of its popular predecessor, namely ns-2.

1.1 Objectives

The purpose of the work done behind this report is to extend the network simulator ns-3 in order to support both the original BATMAN routing protocol as well as the modified version. Using the implemented models in ns-3, the protocols' performances are then to be evaluated in a various range of simulation scenarios. Both versions of the BATMAN protocols are also compared against a well-known and already existing routing protocol in ns-3, namely the Destination Sequenced Distance Vector (DSDV) protocol.

The original goal was to implement all the security mechanisms added to the modified BATMAN protocol. However, due to time constraints, only the elements which were considered the most important was implemented.

1.2 Limitations

Using network simulation to evaluate the performance of network protocols introduces certain limitations by nature. Network simulators are based on statistical models that attempt to resemble real life scenarios as close as possible. They are therefore not always able to capture the true randomness and complexity of the real world. This is further discussed in Chapter 4.

Also, running simulations is a time consuming and resource demanding task. Thus, the amount of various simulations scenarios and simulation runs performed, are adjusted with respect to the time and resources available.

1.3 Method

The work behind this project can roughly be split into four parts:

1. Research and study of background material
2. Implementation of the original BATMAN routing protocol in ns-3

3. Add support for the security extensions to the BATMAN routing protocol in ns-3

4. Use ns-3 to conduct simulations in order to observe, investigate and compare the protocol design, behavior and performance of the different versions

1.4 Document Structure

The remainder of this report is structured in three main parts. Part one contains necessary background material needed to understand the rest of the report such as explanations of the BATMAN protocol as well as the security elements introduced in the modified version. It also gives an insight into to how network simulation works in addition to descriptions of various popular simulators used in research.

Part two of the report describes in detail how the two BATMAN versions are implemented as routing models in the ns-3 network simulator.

The final part three explains how the simulations have been conducted and presents the results retrieved from them. Here we also find a discussion of the results and a final conclusion summing up the report.

Some appendices are also included to give a further explanations and deeper understanding of certain areas. It is remarked in the text if the subject is further described in an appendix.

There is also an appendix which contains a paper based on the work done in this thesis and the work done by Espen Grannes Graarud in his master thesis [Gra11]. The paper was written together with Martin Gilje Jaatun, Dr. Lawrie Brown, and Espen Grannes Graarud and has been submitted to the ICDIM 2011 Conference [1].

[1] http://www.icdim.org/

Part I

Background

Chapter 2

BATMAN - Ad Hoc Routing Protocol

This chapter provides the necessary background material required to understand the basic dynamics behind ad hoc routing as well as the BATMAN routing protocol. This is important in order to understand how the BATMAN protocol is modified for security purposes and how it is implemented in ns-3.

2.1 Ad Hoc Network Routing

In an ad hoc network, it is the participating nodes' duty to maintain and control the communication within the network. This entails that nodes not only send and receive data to and from each other, they must also relay traffic on behalf of other nodes like a router. How the traffic is routed through the network depends on the nodes' routing tables which are maintained by a routing protocol.

Nodes in ad hoc networks are characterized by having the ability to be mobile and still being able to route traffic in the network even though links to other nodes change frequently. Networks where the nodes are mobile are often referred to as Mobile Ad hoc Networks (MANETs).

The mobility of the nodes and the lack of infrastructure create a very dynamic network with a topology that changes randomly and frequently. Due to this unique behavior, regular network functionality such as routing is a challenging task. Classical routing protocols are not well suited for such networks, thus a number of specialized protocols have been developed over the years. The most prominent routing protocols for ad hoc networks are arguably Ad hoc On-Demand Distance Vector (AODV), Destination Sequenced Distance Vector (DSDV), Dynamic Source Routing (DSR), and Optimized Link State Routing (OLSR) [NCÇ$^+$11].

New protocols for ad hoc networks are continuously being designed and amongst these we find the protocol called B.A.T.M.A.N., or BATMAN, which is an abbreviation for "Better Approach To Mobile Ad hoc Networking". Due to its simple routing principles this protocol was chosen to be extended in order to achieve a restricted ad hoc network with access control. The remainder of this chapter describes the BATMAN protocol in further detail.

2.2 The BATMAN Routing Protocol

The development of the ad hoc routing protocol BATMAN was first initiated by a group of developers working on the OLSR protocol. They felt OLSR contained significant shortcomings and that the changes made to it in order to fix them were breaking compatibility with the original protocol as described in RFC3626 [CJay]. Thus, a group of developers decided to design a new and simpler routing protocol called BATMAN that could hopefully become a better alternative to OLSR [Mesayb].

The principle of the BATMAN routing protocol is simple; nodes, or originators as they are also referred to, build their routing tables by flooding the network with routing updates called BATMAN packets. The strategy for a node when finding the best routes in the network, is to determine for each destination one single-hop neighbor which can be utilized as the best next-hop towards the destination [NALWay].

The packet flooding starts with the Originators in the network periodically broadcasting BATMAN packets thus informing their link-local neighbors about their existence (Step 1). Neighbors receiving the packets will rebroadcast them to their own link-local neighbors (Step 2) which will in turn rebroadcast them again to their neighbors (Step 3) and so on. The packets originated from the first node are in this way eventually flooded through the entire network as illustrated in Figure 2.1.

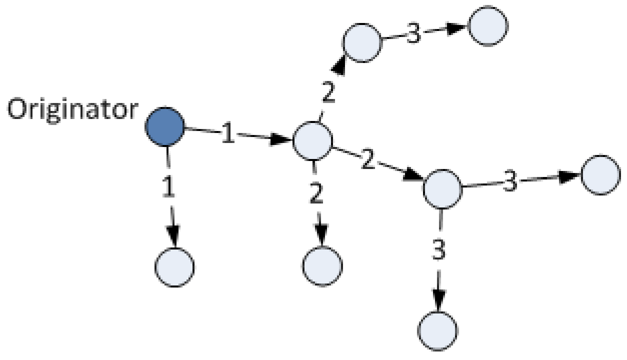


Figure 2.1: Illustration of how a BATMAN packet is flooded through a network.

A BATMAN packet is flooded through the network until every node has received it at least once, or until the packet's Time To Live (TTL) has expired, or until it is lost due to weak communication links. The details of a BATMAN packet are found in Section 2.2.1.

The BATMAN routing protocol has over the years gone through several implementation and testing phases where improvements and changes have been made.

The core routing algorithm has evolved, and at the time of writing the algorithm is at generation IV [NAL07].

The latest generation added three new fields to the BATMAN packet, namely Previous Sender, Transmit Link Quality (TQ) and Host Network Announcement (HNA) Length. These fields where added to improve the protocol's handling of asymmetric links, reduce routing overhead, and enable packet aggregation.

For simplicity the rest of this chapter first explains the BATMAN algorithm III. The last section of the chapter is devoted to describing generation IV which has basically just added functionality on top of the previous algorithm.

2.2.1 Packet Formats

A BATMAN packet consists of one Originator Message (OGM) together with zero or more attached Host Network Announcement (HNA) extension messages. The HNA messages are used when an originator wants to announce that it is connected to another network or host.

However, this feature added by the HNAs is not a critical factor when understanding how the basic BATMAN routing algorithm works. This report focuses on how the performance of this basic routing is affected by the added security extensions explained in Chapter 3. Thus, the HNA messages will not be implemented in the network simulator and therefore not described any further here. However, some more information about these messages can be found in Appendix A.1. Even though the HNA messages are omitted, we sometimes refer to an OGM as a BATMAN packet.

When an OGM is broadcasted, it is encapsulated inside a User Datagram Protocol (UDP) datagram as depicted in Figure 2.2.

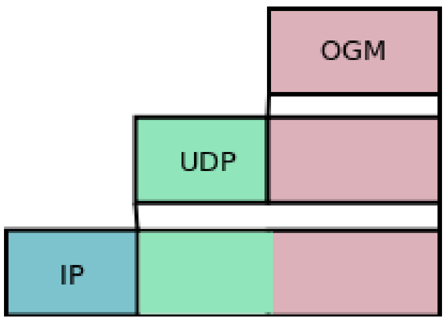


Figure 2.2: A BATMAN packet encapsulated in a UDP datagram.

The OGM contains the most important information used by the BATMAN routing algorithm. The messages have a fixed size of 12 Bytes and an illustration of their general format is shown in Figure 2.3.

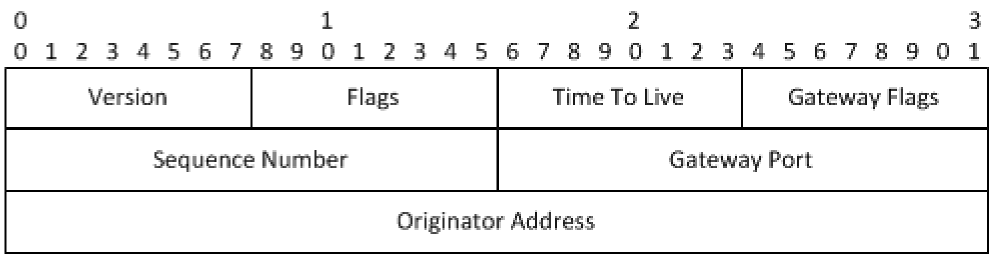


Figure 2.3: Originator Message (OGM) Format.

The different fields in the OGM are shortly explained in the list below:

- **Version**
  Identifies the BATMAN version of the contained message.
- **Flags**
  Bits indicating whether the sender of the OGM is a link-local (direct) neighbor or not.
- **Time To Live (TTL)**
  Is used to limit the number of hops an OGM can do in the network.
- **Gateway Flags**
  Indicates if the node can act as a gateway with access to the Internet.
- **Sequence Number**
  Number added by an originator to every OGM it generates. The number is incremented every time a new OGM is generated.
- **Originator Address**
  The Internet Protocol version 4 (IPv4) address belonging to the originator on which behalf the OGM was generated.

The Gateway Flags are used when a node wants to announce that it can act like a gateway to the Internet. If these flags are set, the Gateway Port field is set to the desired tunneling port of the node acting like a gateway. For the same reasons as for the HNA messages, these fields are also not utilized in the simulations and is thus not described any further. How the remaining fields are used by the BATMAN routing algorithm, is explained in the following sections.

2.2.2 Originator List

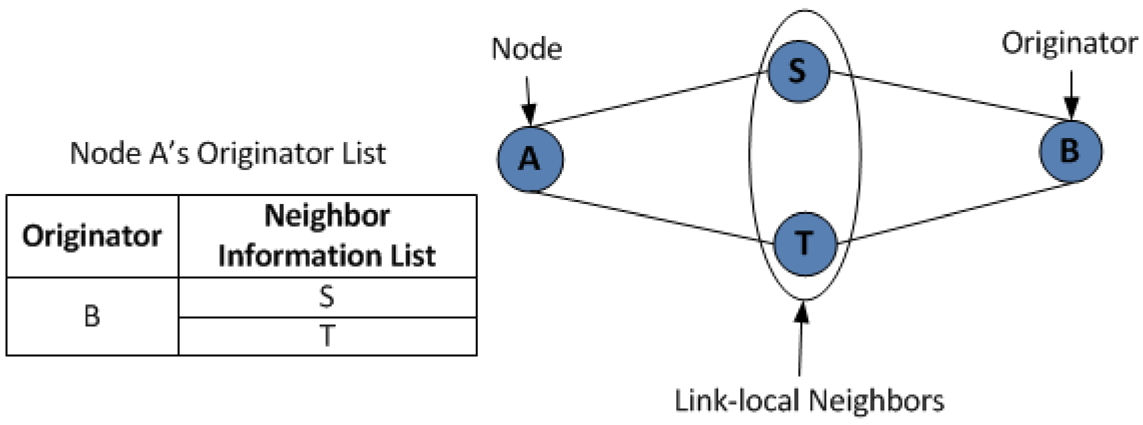
Every node running the BATMAN routing protocol maintains information about all other known originators in a list called the Originator List. The list contains one entry for each originator from which it has received an OGM and is used by the node to choose the best next-hop to a destination.

For each known originator, the most important information a node must maintain in its Originator List, is the following [NALWay]:

- **Originator Address**
  The IPv4 address of the originator as given in the corresponding field of the received OGM.

- **Last Aware time**
  A timestamp that is updated whenever an OGM is received from the given originator.
- **Current Sequence Number**
  This field holds the sequence number from the last accepted OGM from the given originator.
- **Neighbor Information List**
  For each link-local neighbor of the node from which it has received an OGM from the given originator, the following information must be maintained:
  - *Sliding Window*:
    For each in-window sequence number it is remarked if an OGM with this sequence number has been received. Further explained in Section 2.2.3
  - *Packet count*:
    The amount of sequence numbers recorded in the sliding window. This value is used as a metric when choosing the best next-hop to the originator.
  - *Last Valid Time*:
    The timestamp when the last valid OGM was received via this neighbor.
  - *Last TTL*:
    The TTL of the last OGM which was received via this neighbor.

To clarify the concept of the Neighbor Information List; the link-local neighbors of a node via which it has received OGMs originated from another originator in the network, would typically reside in the Neighbor Information List belonging this originator's entry in the node's Originator List. This concept is illustrated in Figure 2.4.


Node A's Originator List

Originator	Neighbor Information List
B	S
	T

Figure 2.4: Illustration of the concept of the Neighbor Information List as seen from node A.

The Originator List functions as a node's routing table and is used when routing traffic in the network. How paths, or best next-hops, are found using the information is this list, is explained in next section.

2.2.3 Sliding Window and Neighbor Ranking

The sequence number in a received OGM is the key information used by the routing algorithm when choosing the best next-hop towards a destination. Here it is the amount of accepted sequence numbers recorded from a link-local neighbor which is used as a metric for the quality of links.

BATMAN utilizes a sliding window to keep track of the most recently received sequence numbers. A sliding window is a set of numbers which slides across the total range of numbers set aside for sequence numbers. When a node receives an OGM, the sequence number in the message is accepted if it resides within this sliding window. More about how the sliding window works can be found in Appendix A.2.

In principle this means that a node chooses the best next-hop towards a destination depending on which link-local neighbor it has received the highest amount of accepted OGMs which originated from the destination. The sliding window makes sure that only the most recently received OGMs are counted and prevents duplicate OGMs to be counted more than once.

Figure 2.5 illustrates this neighbor ranking with node A which has two neighbors, S and T, that can be used as next-hops towards node B. Node A continuously receives OGMs which originates from node B but are rebroadcasted from both node S and T. Node A ranks its neighbors by calculating how many rebroadcasted OGMs (sequence numbers) from node B it has received via each neighbor.

Originator	Neighbors	PacketCount
B	S	27
	T	41

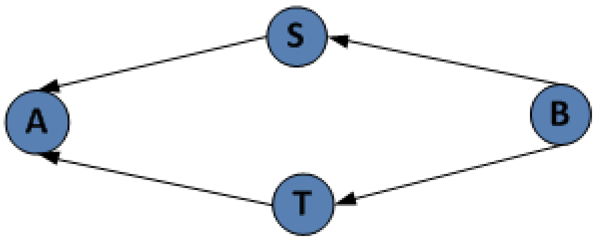


Figure 2.5: An illustration showing the flow of rebroadcasts from node B to node A and a very simplified version of node A's Originator List.

The BATMAN protocol also ensures that the best single-hop neighbor is only selected if it is reachable by a bidirectional link. This is further explained in the next section.

2.2.4 Maintaining the Originator List

Maintaining the originator list involves broadcasting, receiving and rebroadcasting OGMs. To inform other nodes about its presence, an originator generates and broadcasts its own OGMs periodically with its own interface address in the Originator Address field.

Upon reception of an OGM, a node will silently drop the message before further processing depending on some simple rules. Some of these are:

- Version stated in the OGM's version field does not match the nodes own internal version.
- Sender address of the packet is the node's own interface address which means that it's own broadcast has just been echoed back to the node.

If however a node receives an OGM that contains its own interface address in the Originator Address field, it means that the message was originally generated and broadcasted from this node. As illustrated in Figure 2.6, an OGM has been rebroadcasted from neighbor B back to the originator A. This is an indication that the link can be used in both directions, also called a bidirectional link.

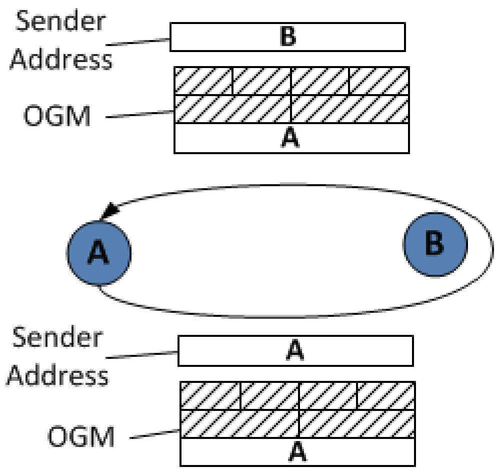


Figure 2.6: An example showing node A receiving a rebroadcast of its own OGM from node B.

Before the OGM is rebroadcasted by node B, some fields must be changed such as the TTL field and the Direct-link flag. All other fields are left unchanged.

If a received OGM is not dropped or a self-generated OGM, a node will rebroadcast it after processing if it was received from a bidirectional link. A node considers a link to a neighbor to be bidirectional if it has received a self-generated OGM with the Direct-link flag set from that neighbor within a reasonable time.

Lastly, if a node has not received any OGMs from an Originator in its Originator List for a time longer than some timeout interval, this entry in the Originator List is removed.

2.3 BATMAN IV

Even though the BATMAN routing algorithm III ensures that only bidirectional links are used when routing packets, the protocol showed significant shortcomings when handling asymmetric links [Mes11]. To improve this weakness the developers decided to add an 8 bit Transmit Link Quality (TQ) field which is explained in Section 2.3.1.

By periodically flooding BATMAN packets, the BATMAN protocol creates a significant amount of overhead in networks that are dense and without heavy packet loss [Mes11]. To reduce the routing overhead the developers introduced the Previous Sender field which ensures that a node does not rebroadcast the same OGM more than once. More about this in Section 2.3.2

The third and last field which was included in the OGM, is the HNA-length. This was added to enable packet aggregation which combines several distinct OGMs into one packet before broadcasting it.

The new OGM format after adding the three new fields is shown in Figure 2.7.

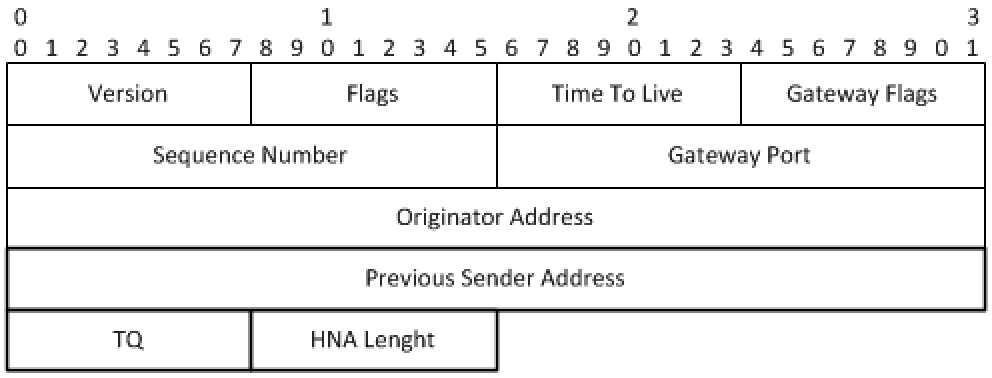


Figure 2.7: Format of the OGM in BATMAN IV.

2.3.1 Transmit Link Quality (TQ)

Link quality in BATMAN IV is divided into two parts, Receiving Link Quality (RQ) and Transmit Link Quality (TQ). The RQ value is the amount of packets a node receives from its neighbors.

BATMAN also keeps track of Echo Link Quality (EQ) which is the number of rebroadcasts of a node's self-generated OGMs received from its direct neighbors as illustrated in Figure 2.8.

Using the values RQ and EQ, a node can derive the local TQ towards a direct neighbor by performing the following calculation:

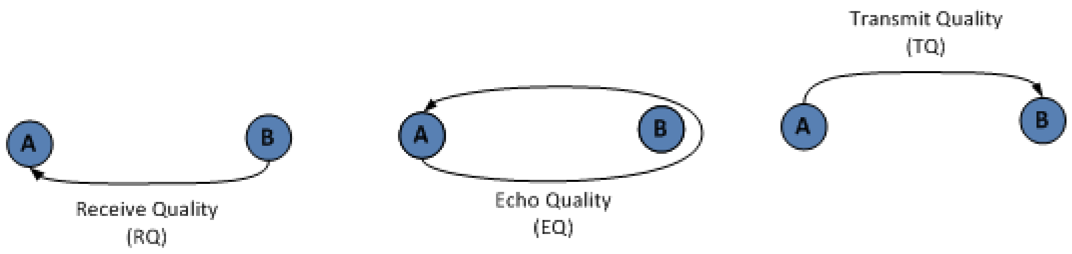


Figure 2.8: Illustration of RQ, EQ and TQ.

$$EQ = RQ \cdot TQ_{local} \Rightarrow TQ_{local} = \frac{EQ}{RQ}$$

This local link quality is propagated through the network to inform other nodes about the transmit quality.

When a node generates own OGMs, the TQ field is set to the maximum length (255) before it is broadcasted. A receiving direct neighbor calculates its own local link quality and adds it to the received TQ value before rebroadcasting the OGM with the new TQ value.

The new TQ value which is put in the OGM is found by performing the following calculation:

$$TQ = TQ_{received} \cdot TQ_{local}$$

An exampel of how the TQ values are calculated is shown in Figure 2.9.

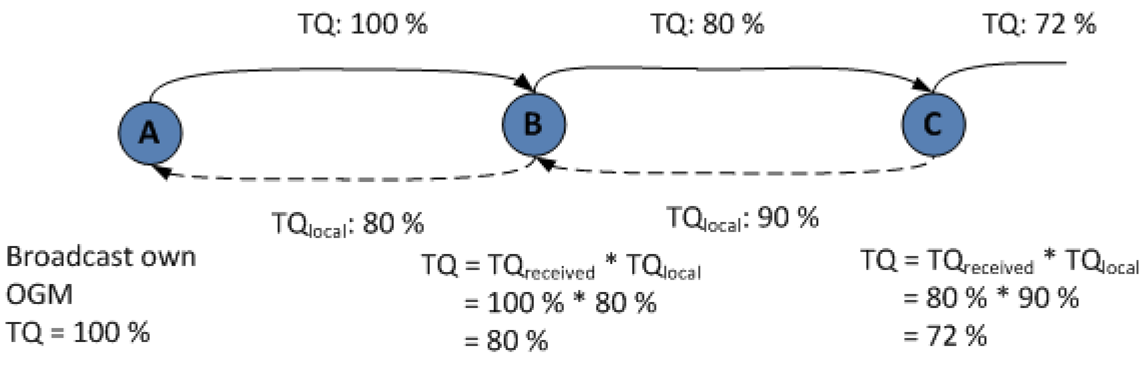

TQ: 100 %
TQ: 80 %
TQ: 72 %
A
B
C
TQ$_{local}$: 80 %
TQ$_{local}$: 90 %
Broadcast own OGM TQ = 100 %
TQ = TQ$_{received}$ * TQ$_{local}$ = 100 % * 80 % = 80 %
TQ = TQ$_{received}$ * TQ$_{local}$ = 80 % * 90 % = 72 %

Figure 2.9: Example of how the TQ in an OGM originated from node A is recalculated in two rebroadcast steps. TQ value of 100% means 255.

2.3.2 Previous Sender

The Previous Sender field was added to reduce echo rebroadcasting which produced unnecessary overhead in the network. This echo cancellation ensures that a node rebroadcasts the same OGM only once after it has received it.

Upon reception of an OGM a node changes the Previous Sender field to the address of the node it received it from before rebroadcasting it. Thus, if a node receives an OGM with its own address in the Previous Sender field, it will silently drop the message.

This echo cancellation is illustrated by an example in Figure 2.10.

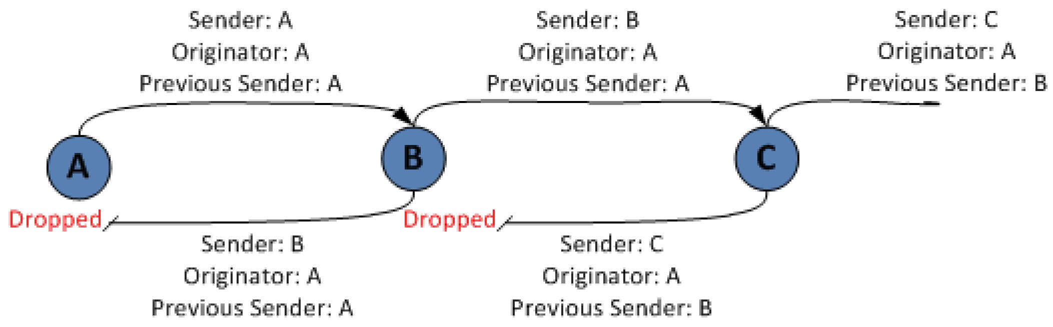


Figure 2.10: An example showing the OGM originated from node A not being rebroadcasted more than once by a node.

Since node B drops the OGM as shown in Figure 2.10, it does not rebroadcast the same message again. Note that node A drops the OGM regardless of the Previous Sender field due to the fact that it is a self-generated OGM.

2.3.3 Asymmetric Link Handling and Hop Penalty

To help ensure that the best bidirectional links are chosen by the BATMAN protocol, an additional value was introduced to penalize links that have poor Receiving Link Quality (RQ).

This penalty is a weighted value and is found with the following function:

$$f_{asym} = (100\% - (100\% - RQ)^3)$$

The penalty has a big influence on the TQ value for links with large packet loss, and only a small influence on links with little packet loss [Mes11].

A node using the BATMAN protocol is only aware of the best next hop towards a destination and not the entire route. Thus, the node does not know how many hops the route may consist of. In some networks it might be desirable to choose the shortest path, as in the minimal amount of hops, to a destination in order to reduce latency and save bandwidth [Mes11]. Therefore a hop penalty was also introduced where every hop an OGM does, decreases the TQ value by a fixed amount.

Both the asymmetric penalty and the hop penalty are added to a TQ value from a received OGM message. The final calculation a node must perform on the TQ value before rebroadcasting the OGM, is as follows:

$$TQ = TQ_{received} \cdot TQ_{local} \cdot f_{asym} \cdot hop\_penalty$$

More details about the BATMAN ad hoc routing protocol, can be found in Appendix A. The next chapter explains how this BATMAN routing protocol is modified in order to include features such as node identification and authentication.

Chapter 3

BATMAN Security Extensions

Now that we have an understanding of how the BATMAN routing protocol works, it is time to investigate how and where it is suitable to introduce security elements to the protocol. The chapter first covers some of the common security issues in ad hoc networks and then continues to describe how the BATMAN protocol is modified in order to include an authentication mechanism.

The security design presented in this chapter is based on a design proposed in a specialization project completed at The Norwegian University of Science and Technology (NTNU) [BG10].

3.1 Security in Ad Hoc Networks

Several challenges and issues are prominent due to ad hoc networks' unique characteristics as discussed in Section 2.1. The issues affect the networks' design, deployment, and performance in addition to how security should be built in. Amongst the major issues relevant for this thesis, we find:

- Resource constraints
- Unreliable medium
- Mobility of nodes

Mobile nodes used in wireless networks usually have limited resources available restricting their computing capacity and battery power [YLY$^+$04]. This affects the choice of cryptography-based security mechanisms which might be computation-intensive to perform.

Also, communication is done on a shared radio channel making it easy for a malicious node to eavesdrop and perform attacks on the network. The communication medium is also very unreliable compared to wired networks, which might result in a high packet loss ratio [NALWay].

Because of ad hoc networks non-hierarchical topology, classical and common security measures are not well suited. Traditional computer networks are infrastructure-based with central entities, such as routers and gateways, which create natural points in the network to add security elements.

Finally, due to the mobility of the nodes, they can frequently join and leave a network at any point in time. If there is no authentication mechanism present, there is no association or relationship between nodes or networks making it easy for an intruder to join a network and carry out an attack.

3.2 Areas of Application

Mobile Ad hoc Networks (MANETs) have a vast range of application areas. The list below names some of the most common situations mentioned in various research papers where MANETs are applicable:

- military operations
- crisis management after a major incident (e.g. war or natural disaster)
- emergency and rescue operations
- wireless sensor network
- collaborative and distributed computing

The protocol design proposed and described in this chapter mainly focuses on being utilized in situations such as or similar to emergency and rescue operations. It is important to consider the area of use for the network in order to consider the possible influences the scenario might have on how security should be applied.

In an emergency or rescue situation, it is natural to have groups of actors with different roles which are in charge of various tasks. Usually there is always a central actor or a central group of actors which have the responsibility of managing and coordinating efforts during the entire operation [Dir07], e.g. the police during a search and rescue operation. New actors arriving to the scene who wants to participate in the operation, would normally have to report to this central administration in order to be assigned a role or delegated a task [Dir07].

With this in mind, the rest of the chapter is devoted to describing the modifications done to the BATMAN protocol and the security measures added. From here on this modified version is also sometimes referred to as ”Secure BATMAN” for short.

3.3 General Concept

The overall goal of the modified BATMAN protocol is to make nodes which are a part of a restricted network, only accept routing updates (Originator Messages (OGMs)) from other nodes that are appropriately authenticated. This entails that they must be in possession of some authentication token which proves their identity.

By only accepting OGMs from nodes that are authenticated, all other traffic sent in the network will only be routed to or via trusted nodes. How this is accomplished is explained in the following sections.

3.4 Proxy Certificates (PCs)

The tokens used to authenticate nodes are Proxy Certificates (PCs) [TET$^+$ay]. These certificates are special versions of traditional X.509 Public-Key Certificates containing a critical certificate information extension.

The extension indicates that the certificate is a Proxy Certificate (PC) and it contains fields such as Proxy Path Length Constraint, Proxy Policy, and Proxy Certificate Path [TET$^+$ay].

The policy field is the part of PCs that makes them beneficial to use as an authentication token in this context. This field enables the possibility of finer granularity of access control by defining rights and restrictions in the network on the node which it is issued to.

Thus, a node may not only be authenticated with its PC, but can also be delegated different restrictions and rights in the network on behalf of the issuer.

3.5 Service Proxy (SP)

The node or entity who signs and issues PCs must have a central role with a higher level of trust than other participating nodes in the network.

Introducing a central entity in a network which is characterized as being infrastructure-less, might seem less than ideal. However, as mentioned in Section 3.2 in the area of use for this kind of ad hoc network, it is common to find a central actor who manages and administrates situations such as an emergency search and rescue operation.

Therefore, it is natural to assign this responsibility of signing and issuing PCs to a central node in the network. We refer to this node as a Service Proxy (SP) and it is in possession of a self-signed PC, called Proxy Certificate 0 (PC0), which it uses to sign other PCs which will be referred to as Proxy Certificate 1s (PC1s).

3.6 Continuous Authentication and Broadcasting of OGMs

As mentioned in Section 3.3, nodes may only accept and process OGMs from other nodes which have been properly authenticated. This means that every OGM broadcasted in the network must somehow prove that it has been sent by an authenticated node and that it has not been altered in transit. This can be accomplished by digitally signing the OGM with the node's private key from its PC0 [RSA78]. Confidentiality of the OGM can also be achieved by encrypting the message with the receiver's public-key.

However, every participating node in a BATMAN network by default generates and broadcasts OGMs every second. In addition they also rebroadcast received OGMs in between their self-generated OGMs, creating a lot of traffic in the network. Thus, signing and encrypting every OGM a node transmits as well as validating every message received, would be computationally infeasible given the nodes' restricted resources as discussed in Section 3.1.

To solve this issue of tying a node's identity to it's OGMs and validating received OGMs without introducing a significant amount of work, the following solution is proposed and explained in the next sections.

3.6.1 Authentication Key Stream

A node generates a symmetric key, referred to as the ephemeral key K, which it unicasts together with a nonce n and an initial value IV to all of its authenticated link-local neighbors[1]. The message is digitally signed for integrity and encrypted with the neighbor's public key from their PC1s for confidentiality, before it is transmitted to the neighbors as shown in Figure 3.1.

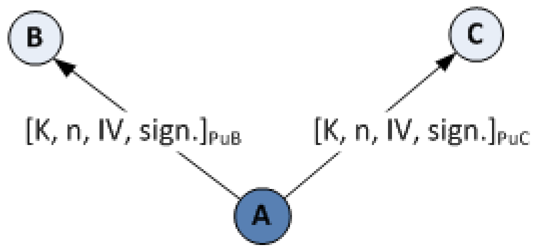


Figure 3.1: An example of node A encrypting the Ephemeral Key K, nonce n, Initial Value (IV), and digital signature sign. with the public-keys of its direct neighbors, $P_U B$ and $P_U C$, and unicasting it to them.

After the transmission, node A and its direct neighbors use the values from the message to generate a key stream using AES-CBC encryption repeatedly. AES-CBC

[1]Entails that it has already received and verified the neighbors' PC1s which is done in the initial authentication phase explained in Section 3.7.1

is the Advanced Encryption Standard (AES) algorithm in Cipher Block Chaining (CBC) mode of operation. This means that a block of plaintext is XORed with the previous cipher text block before it is encrypted with AES [FKGay] as illustrated in Figure 3.2.

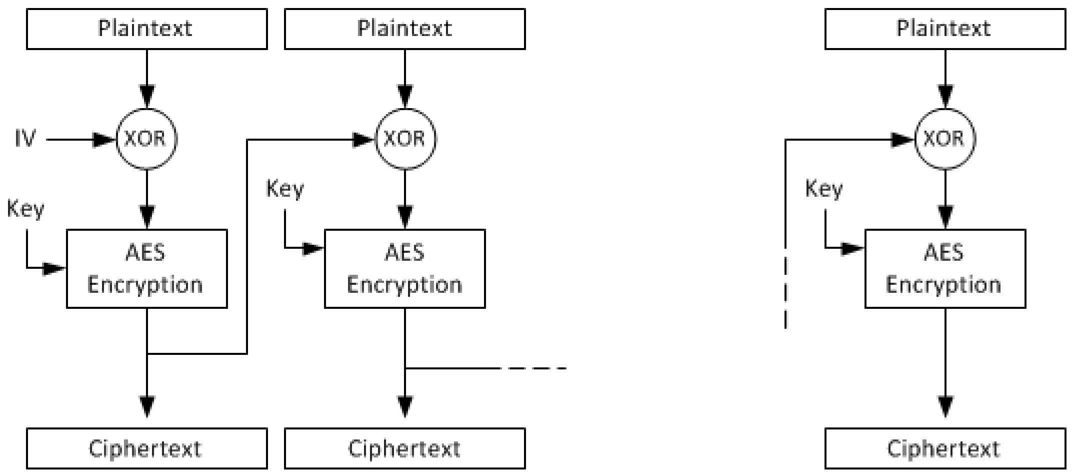


Figure 3.2: An illustration of AES-CBC encryption.

In this case it is the nonce value which is used as the plaintext to be encrypted. In order to generate a large key stream based on the values received, the AES-CBC encryption is repeated where the same nonce value is used but changed for each repetition.

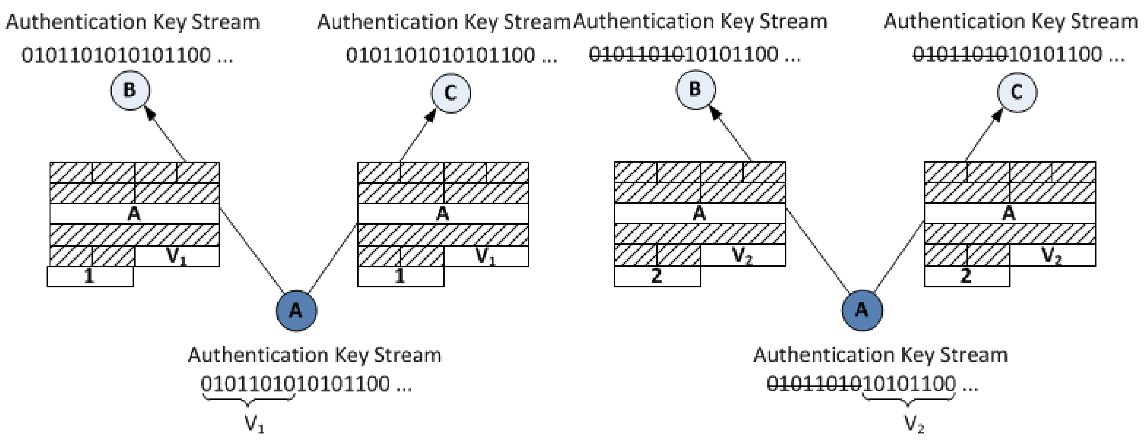
In this way, using the nonce together with the IV and ephemeral key K as input, the AES-CBC encryption creates a chain of cipher text blocks which is referred to as an Authentication Key Stream.

Since all the nodes A, B and C know the same input values, nonce repetition rules, and key stream algorithm, this authentication key stream will be the same for everyone.

As illustrated in Figure 3.3, every OGM which is broadcasted by node A from then on, contains a 16 bit extract of this authentication key stream called a One-Time Password. It also appends a 16 bit sequence number that indicates which part of the key stream the one-time password is taken from.

Upon receiving an OGM from node A, the neighbors B and C verify the one-time password by comparing it to the corresponding authentication key stream they generated themselves.

The neighbors B and C also create their own Ephemeral Keys, nonces and IVs and transmits them to their direct neighbors just as node A. This leads to every node in the network being in possession of their own authentication key streams in addition to one key stream for every direct neighbor they have.



(a) OGMs containing value V_1 and Offset 1. (b) OGMs containing value V_2 and Offset 2.

Figure 3.3: Illustration of node A's OGM transmissions containing extracts of the Authentication Value V.

The point behind generating the key stream of a neighbor, is that it is now possible to verify that future OGMs received from this specific neighbor is actually sent from this neighbor.

Authentication is done hop-by-hop which means that a node only authenticates its link-local neighbors even if an OGM did not originate from them. So, if node B or C are to rebroadcast an OGM received and originated from node A, they replace the one-time password put there by A with their own one-time password from their own authentication key streams. This creates a form of "web of trust" where a node trusts the originator of an OGM if it is trusted by one of its authenticated direct neighbors.

After a certain time interval, the nodes generates new Ephemeral Keys, nonces, and Initial Values and repeat the behavior as explained above. This is to sustain a continuous authentication of the nodes.

To make room for the one-time password and sequence number appended to every OGM sent, two new fields is added as shown in Figure 3.4.

For simplicity, let's refer to both these fields under the term Authentication Fields. If nothing else it specified, when using the abbreviation OGM from now on in this chapter, refers to this modified version of the original OGM.

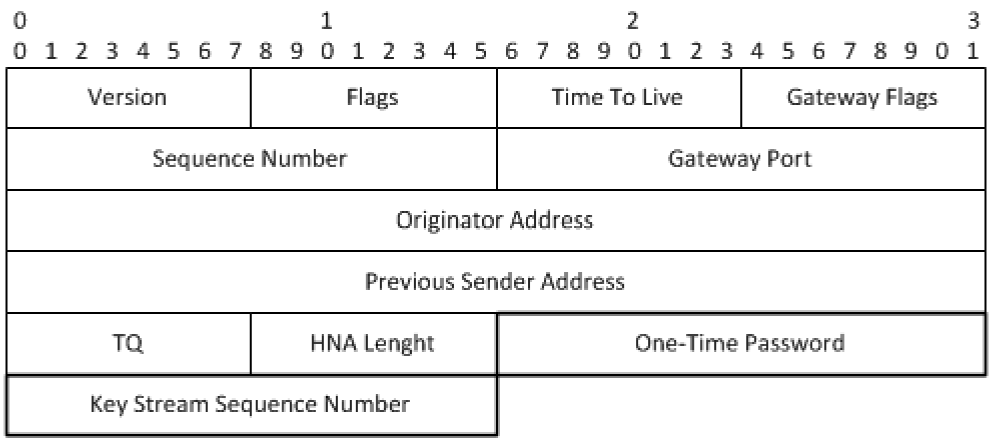


Figure 3.4: Illustration of the OGM format including the fields reserved for security elements.

3.7 PC Signing and Issuing

A node who wants to join a restricted network must be in transmission range of the network's responsible Service Proxy (SP) in order to be issued a PC1.

When not part of any restricted network, a node broadcasts its OGMs as usual, but with empty authentication fields. We refer to this as an empty OGM.

Upon receiving an empty OGM, the SP will engage in a certificate issuing process with the node. This process is similar to how a regular X.509 certificate is issued in traditional networks.

First the SP who received an empty OGM sends an invitation to the originator asking it to create and return an unsigned Proxy Certificate (PC). The originator generates a public-private key pair and sends the public-key in an unsigned PC back to the SP. The SP then signs the certificate with its own PC0 before returning it to the originator. This process in presented as a Message Sequence Chart (MSC) in Figure 3.5.

The originator node is now a part of the restricted network and can be authenticated by all other nodes possessing a PC1 signed by the same SP. How this verification of each others PCs between nodes is performed, is explained in the next section.

3.7.1 Initial Authentication Phase

Before nodes can exchange ephemeral keys and generate Authentication Key Streams, they need to exchange and verify PC1s. This step is called the Initial Authentication Phase.

The phase is triggered if a regular node receives an OGM from a neighbor it has

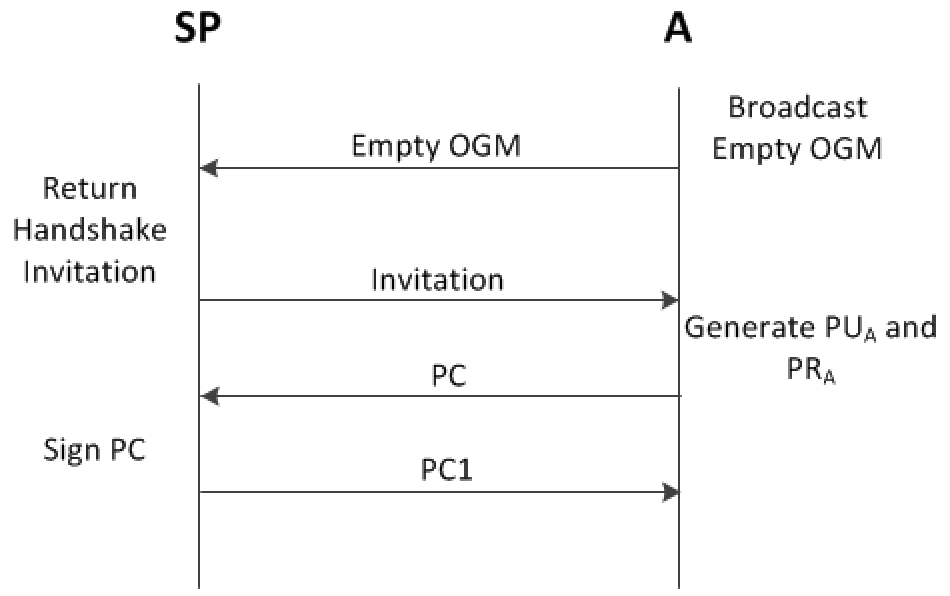


Figure 3.5: Illustration of the OGM format including the fields reserved for security elements.

not yet authenticated. The nodes then continue to exchange and verify each others PC1s which is performed in a process that is similar to how it is done with traditional X.509 certificates.

If the certificates are valid, the nodes exchange ephemeral keys and generate authentication key streams and continue as explained in Section 3.6.1.

If the verification process fails, the nodes will not consider each other as authenticated nodes, thus dropping every OGM sent from them. A node also drops an OGM if the one-time password in the message is not verified.

If a node has verified a neighbors PC, but has not yet received a ephemeral key from it, the node will drop the OGMs until it eventually receives a the necessary information needed from which it can generate an authentication key stream.

3.8 Network Entities

The entities and elements introduced to the BATMAN protocol are summarized and shortly described in Table 3.1.

The nodes also need to keep track of their authenticated direct neighbors including their public-key, ephemeral key K, authentication key stream and last received key stream sequence number. This information is added to the Originator List which is maintained by every node as described in Section 2.2.2.

Entity	**Description**
Service Proxy (SP)	Allowed to sign Proxy Certificates which will be issued to nodes after they have been through an authentication process with the SP.
Proxy Certificate (PC)	Certificate generated by a node that has not been signed yet.
Proxy Certificate 0 (PC0)	Self-signed certificate belonging to a SP. This PC will have the certificate depth of 0 thus we refer to it as a PC0.
Proxy Certificate 1 (PC1)	Certificate that can only be signed by a SP. A node in possession of a PC1 is fully trusted node in the network managed by the SP who signed the certificate.
Ephemeral Key K	A symmetric key generated by every node in the network and unicasted together with a nonce and IV value to every direct neighbor.
Authentication Key Stream	Key stream generated from the AES-CBC algorithm using the ephemeral key, IV and nonce as an input.

Table 3.1: New entities and elements introduced to the modified BATMAN protocol.

Chapter 4

Network Simulation

To get an understanding of the importance of simulating new network protocols, this chapter first explains some basic knowledge about network simulation and its limitations and advantages. The chapter then covers some of the most popular network simulators used in research and education.

4.1 Network Simulation

New network protocols are continuously being designed and developed with the goal of optimizing various operational requirements such as security, reliability, network scaling, mobile networking, and quality-of-service support [BEF$^+$02]. Studying protocols, both individually and interactively, under varied conditions is critical in order to understand their behavior and characteristics.

Building testbeds and labs to study a network protocol can however be both difficult and expensive, especially in large-scale environments. In addition, testbeds and labs are not always capable of reproducing some networking phenomenas, such as wireless radio interference, thus making it difficult to compare and evaluate protocol designs. Lastly they can also be difficult to reconfigure and they have limited flexibility [BBE$^+$99].

Due to the challenges of making real-life models of communication networks, large-scale network simulation has become an increasingly important tool to evaluate protocol design. Various network simulators provide a rich opportunity for efficient experimentation and provide various, but controlled and reproducible network conditions.

Even though network simulators have several advantages compared to testbeds and labs, they also have their limitations and drawbacks. Some of these are discussed in the next section.

4.1.1 Limitations

The goal of simulations is to model real-life systems as closely as possible. To imitate real-life network phenomenas, e.g. propagation loss or node mobility, different mathematical and statistical models need to be created. The models describe the occurrence and behavior of different events and processes.

However, such models can never perfectly resemble the unpredictable behavior of a real network. The precision in the simulation results are only as good as the models used and are thus still only estimates or projected outcomes [BP96].

The details of a simulation is dependent on computer resources and power. Computer limitations, such as memory and processing time, can e.g. limit the number of network objects (nodes, links, and protocol agents) that a designer can simulate [BEF$^+$02].

All simulators adopt some level of abstraction which means that they can have a configurable level of detail for different simulations. Users are with this able to trade simulator performance against level of detail by adjusting the simulator abstraction. However, this introduces a risk of decreasing simulation accuracy when increasing the level of abstraction [BBE$^+$99].

4.1.2 Simulation Scenarios

As mentioned above, different mathematical models define the behavior of a simulation. By applying sets of initial parameters and other variables, the behavior can be changed to attempt to generate a representative simulation scenario.

When generating simulation scenarios the following areas should be assessed [BEF$^+$02]:

- network topologies and dynamics that define realistic models of the movement of mobile nodes
- representative data traffic models
- transmission range and reception power of nodes

Different models and parameters should be chosen such that they create a simulation scenarios which tests the network protocol in the areas desired.

Chapter 6 in this report describes the different models used and parameters changed in order to create representative simulation scenarios used to evaluate the BATMAN protocols.

4.2 Network Simulators

There are several network simulators with varying focuses targeting different areas of research. Some only focus on a particular network type or protocol such as the NIST ATM simulator, while others including ns-2 [BEF$^+$02], ns-3, REAL, Opnet and Insane, target a wider range of protocols [BEF$^+$02].

However, using only one common simulation environment across research efforts can yield many substantial benefits, including [BBE$^+$99]:

- improved validation of the behavior of existing protocols
- a rich infrastructure for developing new protocols
- the opportunity to study large-scale protocol interaction in a controlled environment
- easier comparison of results

Over the past decade, several network researchers have preferred using the network simulator ns-2 when evaluating network systems and protocols [HRFR06]. However, on the basis of some of ns-2's shortcomings and issues, a new simulator, ns-3, has been developed over the recent years. Both simulators are described in the sections below.

4.2.1 ns-2

The network simulator 2, ns-2, is a simulation tool primarily targeted for networking research and educational use. The simulator derives from the old network simulator REal And Large (REAL) from 1989 which was developed with the motivation of studying flow and congestion control schemes in packet-switched data networks [BBE$^+$99, Kesay].

In 1995 the first generation of ns was completed through the VINT project with the hope of becoming a common simulator with advanced features to change the then prominent protocol engineering practices [BEF$^+$02]. The simulator continued to evolve and the second generation, ns-2, was first released in 1996 and was a major architectural change from ns-1 because of its split-level programming model explained next [HRFR06].

ns-2 uses a discrete-event processor as its engine. The simulator provides as mentioned a split-level programming model where the simulation kernel is written in C++ and the simulation setup is done in OTcl, an object oriented version of the scripting language Tcl. The simulation kernel in C++ is responsible for the core set of high-performance simulation primitives while the simulation setup is responsible for the definition, configuration, and control of the actual simulation scenarios [BEF$^+$02].

By utilizing both C++ and OTcl, the simulation maintenance, extension and debugging was separated from the actual simulation itself making it easier to use. In addition the developers and researchers avoided having to recompile the simulator every time a structural change was made. This significantly reduced the total amount of recompilations which were time-consuming in the timeframe the simulator was introduced [HRFR06].

Being an open-source project, the simulator has received substantial contributions from researchers and people who have used the simulator. However, ns-2 is currently only lightly maintained due to the development work on ns-3 which is explained in the following section.

4.2.2 ns-3

As mentioned in Section 4.2, the ns-2 suffered from several issues which prevented the simulators scalability, extensibility and usability. Some of the key issues include:

- **Split-level programming model**
  Few people are familiar to the C++/OTcl structure and the C++/OTcl linking is poorly documented making it a hard to learn and debug. It also puts restrictions on how objects in the simulator may be combined in new ways [HRFR06].
- **Scalability**
  This important property is considered one of the major concerns about ns-2 cited by its users [HRFR06]. The scalability issues is partly due to ns-2's overall software architecture and that it is a sequential execution simulator which creates bottleneck when attempting to simulate more sophisticated communication models e.g. wireless or higher-rate links on a single machine [HRFR06].
- **Realism**
  ns-2 packets are not serialized and deserialized making them unable to function with real-world network systems.
- **Integration**
  ns-2 does not offer many opportunities to integrate with external software, e.g. traffic generators such as iperf and tcplib, or network analysis software such as Wireshark and tcptrace [HRFRay].

The development of ns-3 was initiated in 2006. The main project goals were to develop a simulator that:

- is modular and easily extensible,
- puts more focus on collection of simulation data and statistics,
- has attention to realism and software integration, and

- is flexible

In addition, it was more focus on maintaining an extensive documentation, API [HRFR06].

ns-3 is also a discrete-event network simulator written purely in C++ but with an optional Python interface. Simulation scripts describing simulation scenarios can be written in either C++ or Python.

ns-3 mirrors real network components, protocols and APIs. For instance, a network packet simulated in ns-3 is implemented in detail and is serialized and deserialized when transmitted and received respectively. This increased realism makes the code more reusable, modular, and portable. It also enables the simulator to be used as an emulator in environments including real hardware, software, and networks [HRFR06].

The simulator also provides an extensive tracing system which is a framework that collects statistics by coupling trace sources and trace sinks in the simulator core. The simulator is able to for instance generate packet traces captured in standard formats such as .pcap which can be used together with Wireshark for post-processing [nsnayb].

In order to avoid the issues from which ns-2 suffered from, ns-3 had to break compatibility with its predecessor. This entailed that the relatively large amount of supported protocols in ns-2 could not be directly ported to ns-3, but needed to be re-implemented.

Thus, since the simulator is still fairly new, few protocols and other network applications are supported by the ns-3 at the time of writing. Figure 4.1 shows the different models which are currently implemented in the ns-3 simulator:

However, the simulator is actively supported and the open-source community continuously contributes with new models and helps validate existing ones. Thus, there is a reason to believe that number of supported models will grow in the future.

A more detailed description of the ns-3 architecture is found in Chapter 5 where the implementation of the BATMAN routing models in ns-3 is described.

4.3 Related Simulation Studies

Both ns-2 and ns-3 explained above have been used to evaluate a range of different routing protocols tailored for MANETs [Ver11].

However, support for the BATMAN routing protocol has not yet been implemented

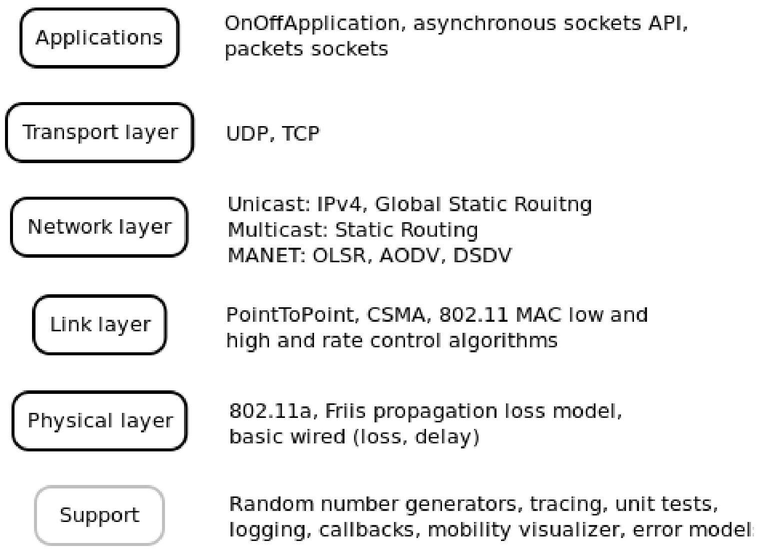

Applications
OnOffApplication, asynchronous sockets API, packets sockets
Transport layer
UDP, TCP
Network layer
Unicast: IPv4, Global Static Rouitng
Multicast: Static Routing
MANET: OLSR, AODV, DSDV
Link layer
PointToPoint, CSMA, 802.11 MAC low and high and rate control algorithms
Physical layer
802.11a, Friis propagation loss model, basic wired (loss, delay)
Support
Random number generators, tracing, unit tests, logging, callbacks, mobility visualizer, error model

Figure 4.1: Some of the different models implemented in the ns-3.10 release of the simulator.

in any network simulators [Mesaya]. Thus no simulations with this protocol have been published at the time of writing.

Part II

BATMAN ns-3 Routing Models

Chapter 5

Implementation

Two new routing models were implemented as routing models in ns-3 including the original BATMAN and a simplified version of the secure BATMAN protocol.

This chapter covers how the ns-3 network simulator is organized and how implementation of these two protocols fit into the simulator's structure. The chapter also describes in more detail how they were implemented including explanations of the most important functions, classes and subclasses as well as their main attributes and components.

The Destination Sequenced Distance Vector (DSDV) protocol is the newest ad hoc routing protocol added to the latest stable release ns-3.10 at the time of writing [NCÇ+11]. The ns-3 model of this protocol was used as a source of inspiration when implementing the BATMAN routing protocol in ns-3. The implementation was also inspired by studying the real-life BATMAN protocol and its RFC draft [NALWay].

5.1 ns-3 Architecture

Figure 5.1 gives a schematic view of how the source code in ns-3 is organized. It shows that the simulator is divided into separate modules where the modules only have dependencies to other modules placed beneath them. A ns-3 module is built as a separate software library where ns-3 programs can link the modules they need in order to conduct their simulations. A module may consist of one or more models which are abstract representations of real-world objects, protocols, devices and so on [nsnaya].

The core of the simulator consists of the three modules `core`, `common` and `simulator`. Together they create a generic simulation foundation which is common across all protocol, hardware, and environmental models making it usable for any kind of network, not just only IP-based networks.

The two modules in the top layers of Figure 5.1, `helper` and `test`, are supplements to the C++ core API. ns-3 programs, also referred to as simulations scenarios or

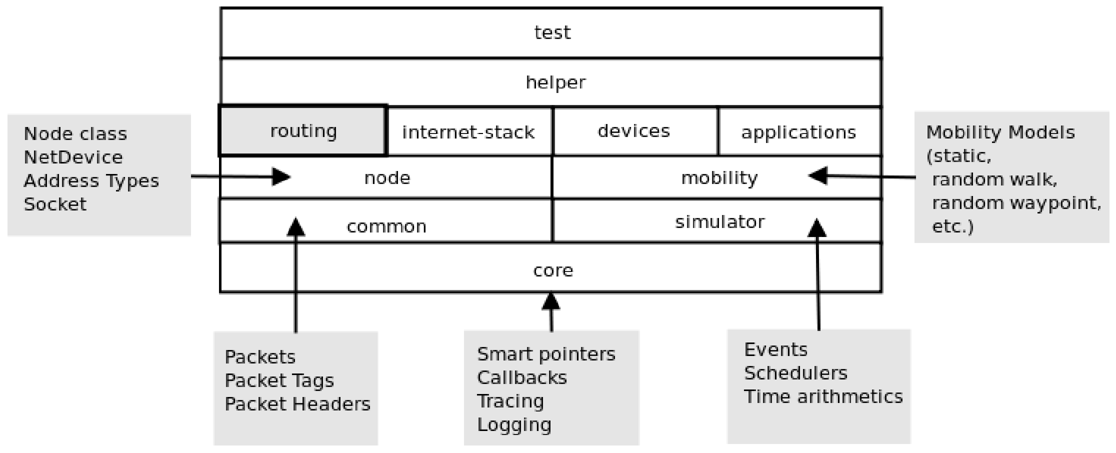


Figure 5.1: Software organization of the ns-3 simulator, based on the figure from [nsnayb].

scripts, can access the core API directly or use the high-level wrappers and encapsulations found in the `helper` module [nsnayb].

The other layers in Figure 5.1 add the networking-specific components of ns-3. For instance, the `internet-stack` module includes implementations of protocols such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) while the `routing` module provides different models of Internet Protocol version 4 (IPv4) routing protocols.

The organization of ns-3 provides a modular source code where different models can be added to a module without having to make changes to other modules or the ns-3 structure. The BATMAN routing protocol was implemented in the `routing` module which is highlighted in Figure 5.1. A BATMAN `helper` class was also implemented to assist when using the BATMAN protocol in simulation scenarios.

The remainder of this chapter first explains how the original BATMAN protocol was implemented as a routing model in ns-3. It then describes how this model was used as a base when implementing a routing model for the simplified Secure BATMAN protocol.

5.2 Class Interaction

The relations between the BATMAN classes implemented in the `routing` module, are presented in Figure 5.2.

As the figure illustrates, `ns3::batman::RoutingProtocol` is a subclass of the already implemented abstract base class `ns3::Ipv4RoutingProtocol`. This class performs the main routing tasks and contains functions that connect the routing model to the rest of the ns-3 core.

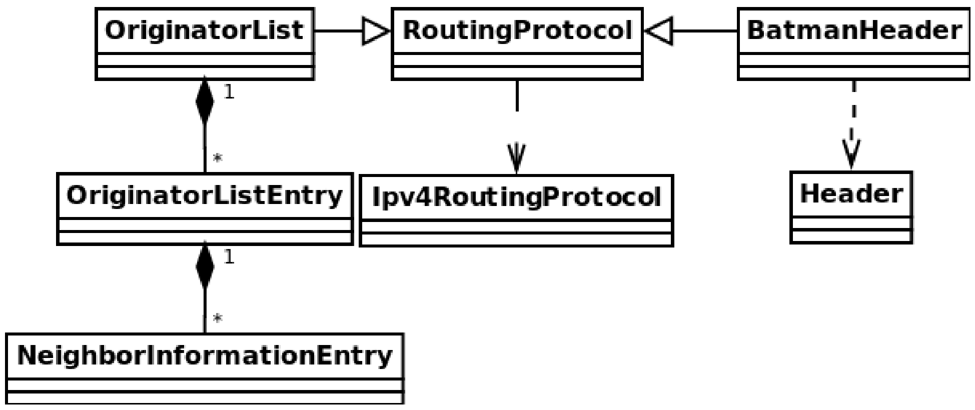


Figure 5.2: Class Diagram for the BATMAN routing model in ns-3 omitting attributes and methods due to size issues.

The `ns3::batman::OriginatorList` class contains a collection of originator list entries declared in the `ns3::batman::OriginatorListEntry` class. These entries are equivalent to the originator list entries as explained in Section 2.2.2. Every originator list entry also has a neighbor information list which includes a set of neighbor information entries declared in the `ns3::batman::NeighborInformationEntry` class.

The `ns3::batman::BatmanHeader` class implements the packet format and it is extended from the existing `ns3::Header` base class.

Two additional classes are also implemented in the BATMAN routing model, namely `ns3:batman:bitarray` and `ns3:batman:ringbuffer`. These classes only contain functions used to update the sliding windows belonging to neighbor information list entries. Since they do not have any other tasks in the routing model they are omitted from the class diagram in Figure 5.2 and not described in more detail. The rest of the classes mentioned above are further explained in the following sections.

5.3 Originator Message (OGM)

The `ns3::batman::BatmanHeader` class defines the format of the BATMAN packet which consists of one OGM and zero or more optional Host Network Announcement (HNA) extension messages. But, as discussed in Section 2.2.1, the HNA feature is not used during the simulations and is therefore not implemented in ns-3.

The OGM without the optional trailing HNA messages, was implemented after the format from BATMAN IV explained in Section 2.3 and is illustrated in Figure 5.3.

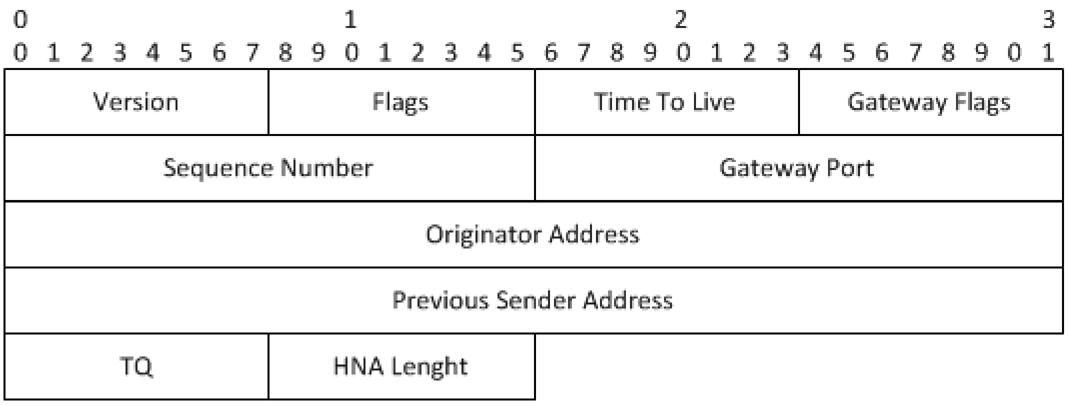


Figure 5.3: Format of the OGM excluding the optional HNA messages.

The fields which are not used, Gateway Flags, Gateway Port, and HNA Length, are not omitted in the implementation, just set to zero when running the protocol.

The `ns3::batman::BatmanHeader` class includes two important methods inherited from the abstract base class `ns3::Header`, namely `Serialize` and `Deserialize`. These methods convert the BATMAN header into a byte buffer in its network representation and makes it possible to use the routing protocol in simulations which interact with real-life networks as discussed in Section 4.2.2.

5.4 Originator List

A node's Originator List is equivalent to an ordinary routing table and it is defined in the `ns3::batman::OriginatorList` class as a C++ map. This class includes methods to add, delete, update, purge and look up entries in the Originator List.

A node stores an entry in its Originator List for every originator from which it has received an OGM just as described in Section 2.2.2 and 2.2.4. These originator list entries are defined in the `ns3::batman::OriginatorListEntry` class and holds the following information:

- **Originator Address**
- **Last Aware Time**
- **Last Time To Live**
- **Current Sequence Number**
- **Next Hop**
- **Neighbor Information List**

Every originator in a node's Originator List may be reached via several link-local neighbors. These neighbors are added as entries in the Neighbor Information List which is a C++ map called `m_neighborInformationList`. The entries in the list are declared in `ns3::batman::NeighborInformationEntry` class and contains the following attributes:

- **Neighbor Address**
- **Last Valid Time**
- **Last Time To Live**
- **Sliding Window for rebroadcasts of own OGMs (Echo Link Quality (EQ))**
- **Sliding Window for total received OGMs from this neighbor (Receiving Link Quality (RQ))**

The information in this list is used by the protocol to choose the best next-hop towards the originator to which the list belongs as described in Section 2.2.3.

5.5 BATMAN Routing Protocol

The main routing logic is performed in the `ns3::batman::RoutingProtocol` class making it the most important component in the BATMAN routing model. The class inherits functions from its parent class such as `RouteInput` and `RouteOutput` which tie together the network layers in the simulator. It also combines functions from the other classes in the model to build and maintain a node's Originator List.

The main functions in the `ns3::batman::RoutingProtocol` class, are:

- `BroadcastOriginatorMessage`
- `RecvOriginatorMessage`
- `ReBroadcastOriginatorMessage`

The function `BroadcastOriginatorMessage` is in charge of periodically broadcasting OGMs on behalf of a node as explained in Section 2.2. Is keeps a timer called `OriginatorIntervalTimer` which triggers the periodic broadcasts of an OGM.

`RecvOriginatorMessage` handles the processing of received OGMs. Here an OGM first goes through some initial checks verifying its Version field, Previous Sender field and so on. If all checks are passed, it is then investigated whether the received OGM is a self-generated OGM rebroadcasted from a neighbor. If this is the case, the function `CountRebroadcast` is called which updates this neighbor's EQ value which is the amount received rebroadcasts of self-generated OGMs as explained in Section 2.3.1.

Two other important checks performed is found in the functions `BidirectionalCheck` and `DuplicatePacketCheck`. `DuplicatePacketCheck` examines if the OGM has already been received by checking if the sequence number in the OGM has been marked in the corresponding sliding window. If not, the sliding window is updated and the RQ value of the neighbor who sent the OGM is updated.

`BidirectionalCheck` uses both the RQ and the EQ value to check if the sender of the OGM is a bidirectional neighbor. If the neighbor turns out to be bidirectional, the calculations described in Section 2.3.1 is performed and a new Transmit Link Quality (TQ) value is calculated. This new value is put in the OGM if it is to be rebroadcasted.

If the OGM is both received from a bidirectional neighbor and is not a duplicate packet, the function `UpdateOriginator` is called. This function updates the neighbor ranking choosing the best link-local neighbor as the next-hop towards the originator of the OGM.

Finally, the OGM is rebroadcasted if it has passed all the checks and has a Time To Live (TTL) value more than zero. The function `ReBroadcastOriginatorMessage` changes the appropriate fields in the OGM before it rebroadcasts it.

5.6 Routing Attributes and Default Values

Table 5.1 shows some important attributes and their default values used in the implementation.

Attribute	**Defaults**	**Description**
`DEFAULT_VERSION`	4	BATMAN version
`MAX_SEQUENCE_NUMBER`	65535	Maximum sequence number allowed
`DEFAULT_TTL`	50	Default Time To Live value
`OGM_BROADCAST_INTERVAL`	1 s	Periodic interval between broadcasting of OGMs
`TQ_LOCAL_WINDOW_SIZE`	64	Size in bytes of the sliding window used when calculating TQ values.
`PURGE_TIMEOUT`	200 s	Lifetime of an originator list entry which has not been updated recently.
`HOP_PENALTY`	5	Penalty added to an OGMs TQ value for every hop made in the network.
`MAX_TQ_VALUE`	255	Maximum TQ value possible.

Table 5.1: BATMAN attributes and their default values.

5.7 Protocol Validation

As mentioned in Section 4.3, the BATMAN routing protocol has not been implemented in any network simulator before. Thus, in order to validate the protocol it had to be compared against its real-life counterpart. Detailed debug output from a simple simulation and a small real-life network was studied to ensure that the protocols behaved identically. Later it was also compared against a well-known routing protocol, DSDV, during various simulations as explained in Chapter 6.

5.8 BATMAN Security Extensions

The newly implemented routing model supporting the original BATMAN protocol was used as a base when implementing the second routing model for the Secure BATMAN protocol. As mentioned in Chapter 1, parts of the Secure BATMAN protocol was not implemented in the ns-3 routing model due to time constraints. Even though the ns-3 network simulator provides a modular simulator core which is easily extensible, it is still a complex tool which takes some time to learn. In addition, the time taken to validate the routing model for the original BATMAN took longer than expected. Thus, only the most important elements of the Secure BATMAN routing protocol was implemented in the ns-3 model.

5.8.1 Simplified Secure BATMAN

The main focus when implementing a simplified version of Secure BATMAN was to try capturing the security elements which affects the total routing performance the most. The added work which the Secure BATMAN imposes on the original protocol can roughly be summarized into three main steps:

1. Proxy Certificate (PC) signing and issuing,
2. verification of other nodes' PCs, and
3. continuous authentication of received OGMs

Assuming that a node is issued a Proxy Certificate 1 (PC1) which is valid during an entire emergency and rescue operation for instance, this first step is performed only once at the very beginning of the operation.

Also, the exchanging and verification of other nodes' PC1s is also done once. After a node's PC1 has been verified, the nodes start to exchange authentication material packets and generating key streams as explained in Section 3.6.

Thus, it is this continuous authentication of OGMs which probably has the most continuous impact on the total routing performance of the protocol. The implementation of the routing model therefore prioritized on including the functionality where authentication material is created and shared as well as the continuous verification of one-time passwords in received OGMs.

5.8.2 Cryptographic Functions and OpenSSL

The cryptographic functions added to the routing model were based based on and inspired by the work done by Espen Grannes Graarud on his master thesis at The Norwegian University of Science and Technology (NTNU) [Gra11]. His thesis involved developing a real-life implementation of the Secure BATMAN protocol.

The functions are created using the OpenSSl cryptographic library. This is a library containing implementations of the industry's best-regarded algorithms, including encryption algorithms such as Advanced Encryption Standard (AES) and RSA as well as message digest algorithms and message authentication codes [VMC02].

The following sections describe the new classes added and the most important functions and attributes included in the Secure BATMAN ns-3 routing model.

5.8.3 Packet Format

As explained in Section 3.6.1, the OGM was extended by adding two new fields in order to include the authentication mechanisms. The format is shown in Figure 3.4.

Similar to the original BATMAN implementation, the the packet format is defined in `ns3::securebatman::SecureBatmanHeader` which is extended from `ns3::Header`.

In addition to the `ns3::securebatman::SecureBatmanHeader`, a new class called `ns3::securebatman::SecureAmHeader` is also implemented in the routing model. This class defines the Authentication Message (AM) which is used to share key stream material used to generate key stream. How the AM is used is explained in the next section and the format of the AM is shown in Figure 5.4.

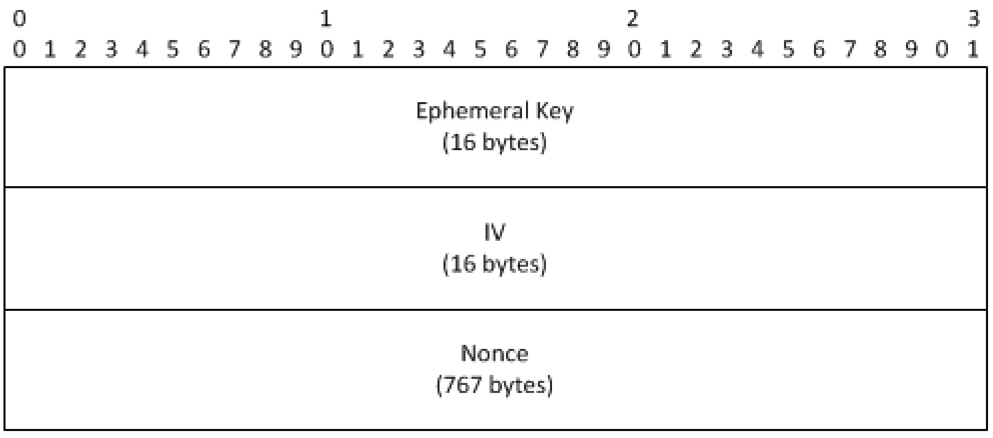


Figure 5.4: Illustration of the Authentication Message (AM) used to share key stream material with neighboring nodes.

5.8.4 Key Stream Material and Key Stream Generation

The most important functions added in this model is:

- `GenerateKeystreamMaterial`
- `GenerateKeyStream`
- `GenerateNeighborKeyStream`

- `CheckOneTimePassword`

The first step done by a node running the secure BATMAN protocol, is to generate its own ephemeral key, nonce and Initial Value (IV) which it transmits in an AM to its link-local neighbors. After the transmission it generates its own authentication key stream and continues to broadcast regular OGM containing one-time passwords from this key stream.

All the functions used for cryptographic purposes such as ephemeral key generation are defined in the new class called `am`. The functions residing in this class use tools from the OpenSSL library as mentioned in Section 5.8.2.

If a node receives an AM for the first time from a link-local neighbor, it retrieves the values from the message and generates the neighbors authentication key stream by using the function `GenerateNeighborKeystream`. For every OGM received from now on from this specific neighbor, the function `CheckOneTimePassword` is used to verify the one-time password in the OGM.

A timer called `m_amPacketIntervalTimer` was also added to trigger the generation of new key stream material to be transmitted to neighbors. This is to sustain continuous authentication as explained in Section 3.6.1.

Part III

Simulations and Results

Chapter 6

Simulation Setup

The previous chapter explained how both versions of the BATMAN protocol are implemented as routing models in ns-3. This chapter describes the different scenarios simulated in ns-3 using these routing models, as well as the different metrics that were measured during the simulations.

6.1 Performance Metrics

The main goal of the simulations is to measure both the original and the modified BATMAN protocol's ability to react to network topology changes while still continuing to successfully deliver packets efficiently to a destination.

This ability can be defined by a selection of performance metrics which is measured during the simulations and then analyzed and compared. The following metrics are valuable when evaluating the performance of a routing protocol [CBD02, BMJ$^+$98]:

- **Packet Delivery Ratio (PDR):** The amount of packets received divided by the amount of packets actually sent by the application layer.
- **Routing Overhead:** The total amount of routing packets (OGMs) measured in bytes transmitted during the entire simulation.
- **Packet Delay:** The time taken by a packet when transmitted from a source to a destination measured at the MAC layer.

The PDR gives an indication of the protocol's loss rate which affects the maximum throughput that the network can support. Routing overhead shows the scalability of the protocol while the transmission delay indicates how efficient the protocol is when choosing the best path to a destination. All these metrics will be monitored and measured during the simulations conducted.

6.2 Mobility Models

Mobility models describe the movement of a node during a network simulation. This includes the direction of movement, velocity and acceleration of the node over time.

Several mobility models exist in ns-3 and the most relevant for the simulations performed in this report, are [CBD02]:

- **Random Walk Mobility Model:** Describes mobility patterns were a node moves from its current position to a new random destination moving with random speed.
- **Random Waypoint Mobility Model:** Mobility model which also includes pause times between the changes in destination and speed.
- **Random Direction Mobility Model:** Describes node movement patterns where the nodes are forced to move to the edge of the simulation area before changing direction and speed.

The models mentioned above are all entity mobility models which define the movement of the nodes independently of each other. The two first models are the most common mobility models used by researchers [CBD02].

The mobility model utilized in the simulation scenarios in this report, is the Random Waypoint Mobility Model. Even though it is one of the most commonly used models in ad hoc research, it is proved that it suffers from certain issues which might affect the performance results from simulations [BRS03, CBD02]. However, this issue can be avoided by following some simple recommendations. One of these recommendations is to discard the initial simulation time produced by the model [CBD02]. Thus, a small ”settling time” is introduced before traffic is sent in the network in order to hopefully reduce the potential impact of this issue.

6.3 Methodology and Simulation Setup

Running a simulation in ns-3 works by writing a program which describes a simulation scenario including the different combinations of models used from the ns-3 core, e.g. node movement model, Internet-stack model, and traffic generation application.

By changing the parameters belonging to the models used, different variations of the scenario can be made in order to test different aspects, e.g. changing the pause time parameter in the Random Waypoint Model.

The different parameters, simulation scenario and methodology of how the simulations were executed, were inspired by previous studies involving simulations with other ad hoc routing protocols [NCÇ+11, BMJ+98, DPR00].

6.3.1 Physical Space and Node Movement

The simulations are performed using a varying amount of nodes, 10, 20 and 30, which are moving in a rectangular flat space (1500x300m^2).

The nodes movement is as mentioned described by the `RandomWaypointMobility` model already implemented in ns-3. The nodes' velocity is set to vary between 0 and 20 m/s during a simulation run and every run is done with 10 different pause times varying from 0 to 900 seconds.

The total simulation time for every simulation run is set to 900 seconds. This means that a simulation scenario running with pause time 0 seconds entails that nodes are moving continuously with no stops. When using a pause time of 900 seconds, the nodes stay at a fixed position the entire simulation run.

6.3.2 Traffic Generation and Flows

All nodes transmit and receive constant data traffic to and from other nodes in the network. Based on the previous studies, a packet size of 64 bytes was chosen due to very low PDR values when using a larger packet size (>500 bytes) [NCÇ+11, BMJ+98].

The nodes are configured to generate Constant Bit Rate (CBR) traffic using the `OnOff` application in ns-3 with a data rate of 4 packets/s (256 Bps). The simulator is configured to create as many traffic flows as the amount of nodes in the network and it is ensured that every node both receives and transmits data.

Nodes generate CBR traffic using 802.11b MAC over the Friis propagation loss model [Fri46] to limit the nodes transmission range [NCÇ+11, BMJ+98].

All the simulation parameters that are constant for every simulation run, is summarized in Table 6.1.

Parameter	**Value**
Node Velocity	0 - 20 m/s
Packet Size	64 bytes
Data Rate	256 Bps
Settling Time	30 s
Simulation Time	930 s
BATMAN OGM Broadcasting Interval	1 s

Table 6.1: Simulation parameters that stay constant for every simulation run.

6.3.3 Simulation Statistics and Data Collection

The ns-3 tools `DataCollector` and `DataCalculator`, are used to count and gather data and statistics from different trace sources and sinks couplings in the ns-3 core.

The data collected is averaged over 10 repeated simulation runs to get an central tendency of the data set. This sums up to about 300 simulation runs for one routing protocol with different variations of the simulation scenario as illustrated in Table 6.2.

Repetitions	**Node Count**	**Pause Times (s)**
10 x	10	0
		100
		...
		800
		900
	20	0
		100
		...
		800
		900
	30	0
		100
		...
		800
		900

Table 6.2: Illustrations of the different parameter combinations used and amount of repetitions of the simulation runs.

After a simulation run has completed, the `DataCollector` is in charge of storing the data in a SQLite database [SQLay] for post-processing and analysis.

Due to some memory allocation issues when running simulations with the Secure BATMAN routing model, this protocol was not simulated with a node amount higher than 10.

6.4 Running the Simulations

The simulations are controlled by a simulation script which launches the different simulation scenarios with the different parameter combinations. It is also in charge of querying the SQLite database and parsing results for creating graphs with Gnuplot [BCC$^+$ay].

The order of the steps performed by the simulation script is described in the list below and illustrated in Figure 6.1.

1. Start simulation script
2. Script runs the simulation scenario written in C++ specifying the different parameters to be used (pause time, amount of nodes etc.)

3. Simulation scenario measures and store data from the simulation in the SQLite database

4. Simulation script queries the database and store data points in a file

5. Simulation script invokes Gnuplot to create the graphs from the data points received and parse from the database

The Figure 6.1 illustrates the workflow of the simulation script.

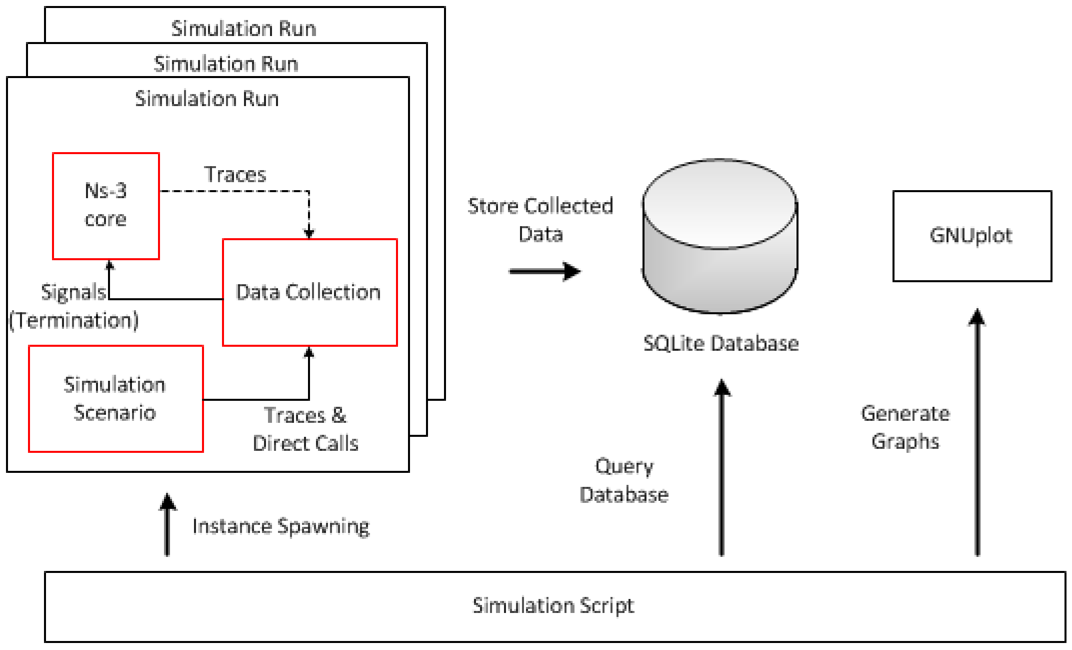


Figure 6.1: Workflow and involved entities during the lifetime of the simulation script.

The simulation script can be found in Appendix C.

6.5 Simple Performance Comparison of DSDV and BATMAN

BATMAN's main principle is that routing information is flooded through the network and thus shared amongst every single participating node. The flooding is achieved by every node broadcasting routing information which is rebroadcasted through the network until all the other nodes have received the information. This creates a lot of traffic and increases the possibility of packet collisions and interference. Also network scalability becomes an issue.

The developers justify this design choice with the fact that wireless networks are by nature very lossy and a high packet loss is expected [NALWay]. Thus, the simplest and most efficient way of maintaining the network, is to simply flood it with routing information and expect that a large majority of the packets will be lost.

DSDV is also a proactive routing protocol, but it produces however way less routing overhead compared to BATMAN. Therefore, it would be interesting to also investigate how the performance of this protocol is compared to BATMAN when exposed to a more lossy wireless environment.

This comparison was done by performing the same simulations as explained above, but decreasing the nodes' transmission power to reduce their transmission range. This creates weaker links between nodes making a more unstable network. Thus, the protocols were tested in what is referred to as one ”strong network” and one ”weak network”.

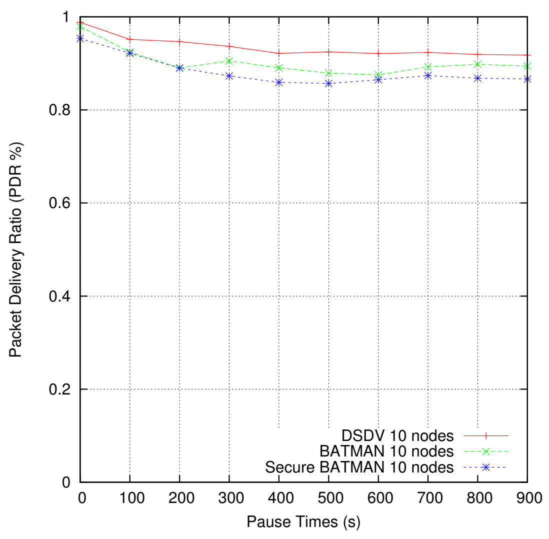
The results from the simulation scenarios described, are presented in the next chapter.

Chapter 7

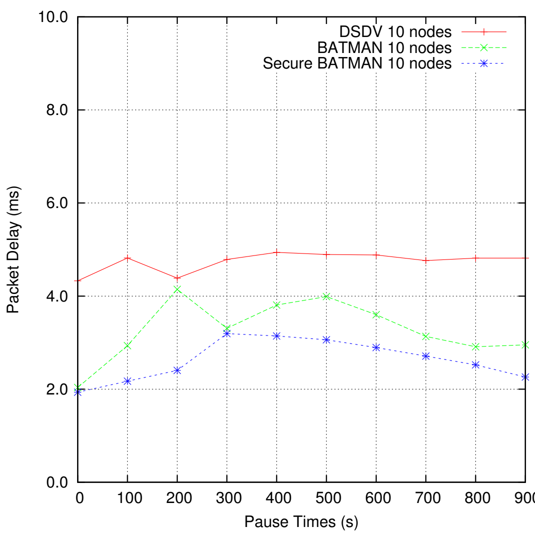
Simulation Results

This chapter presents the results from the different variations of the simulation scenario described in the previous chapter. In general the results show that the Secure BATMAN protocol performs at the same level as both the original BATMAN and Destination Sequenced Distance Vector (DSDV). The additional simulations conducted with the original BATMAN protocol and DSDV indicate that BATMAN does not show signs of outperforming DSDV in a ”weak network”.

Figure 7.1 shows Packet Delivery Ratio (PDR) and packet delay results from the simulations running the Secure BATMAN, BATMAN and DSDV with 10 nodes and 10 traffic flows.



(a) PDR with varying pause times.



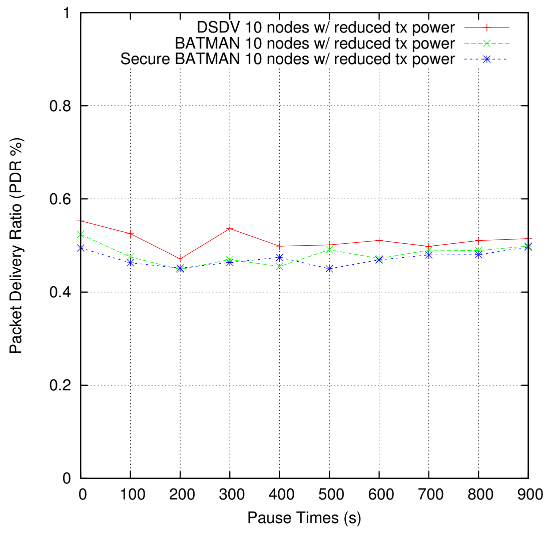
(b) Packet delay with varying pause times.

Figure 7.1: Results from BATMAN, Secure BATMAN and DSDV running with 10 nodes and 10 source and sink pairs.

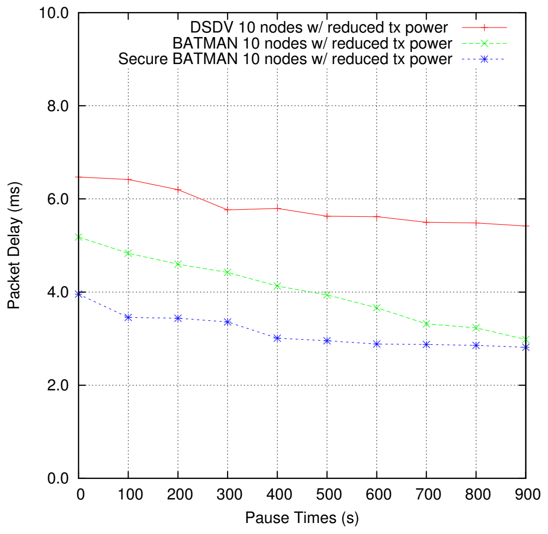
As seen from the graphs in Figure 7.1a, the PDR values of all three routing protocols stay well above 80%. Interestingly, the Secure BATMAN protocol’s PDR also stay at approximately the same level as the two other protocols. At pause time zero, which is equivalent to continuous node movement, all three protocols show their best behavior with the highest PDR values.

When looking at the average packet delay in Figure 7.1b it is surprisingly the Secure BATMAN protocol which has the lowest values.

As mentioned in Section 6.5, the same simulations were run with decreased transmission power in order to create a weaker network. The Figure 7.2 shows the PDR values and packet delays derived from the simulations results.



(a) PDR with varying pause times.



(b) Packet delay with varying pause times.

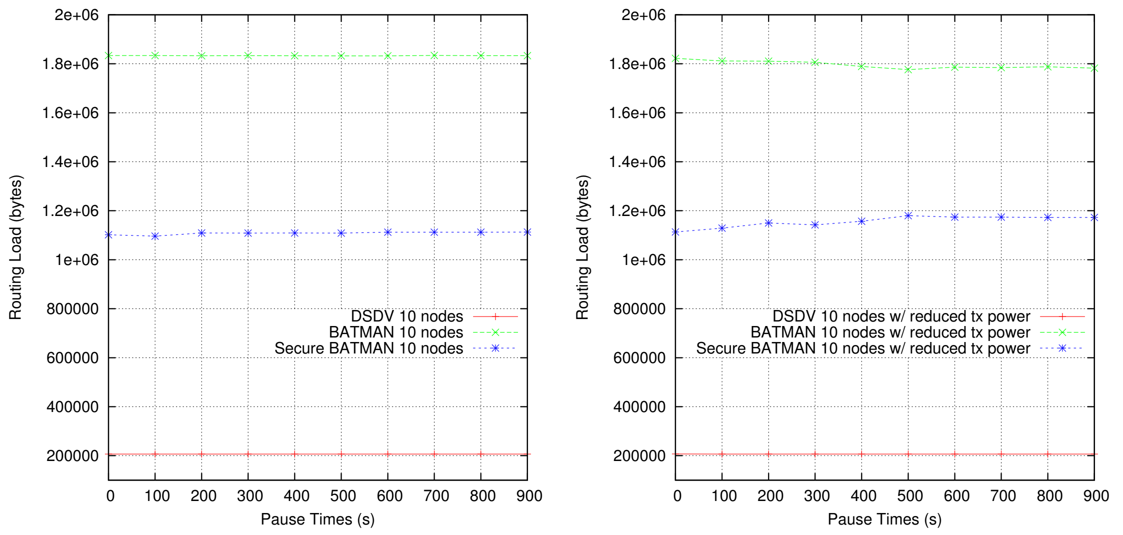
Figure 7.2: Results from BATMAN, Secure BATMAN and DSDV running with 10 nodes and 10 traffic flows where the transmission power has been reduced.

From reducing the transmission power the PDR values drop significantly as shown in Figure 7.2a. This is due to the fact that packets no longer reach as far as in the previous scenario and the routing overhead create more collisions and interference since the signals are weaker. Still all three protocols preform almost equally well at delivering packets from source to destination.

The packet delays of all the three protocols presented in Figure 7.2b, are slightly increased in this scenario. This is natural as the packets probably have to use longer paths (more hops) to arrive at the destination since the signals are weaker. Still it is the Secure BATMAN protocol which has the lowest average packet delays.

Figure 7.3 shows the routing load in bytes produced by the three protocols during both of two first scenarios, with and without reduced transmission power.

As expected, both BATMAN and Secure BATMAN create way more load on the network compared to DSDV. However, all three protocols create an approximately



(a) Routing overhead with varying pause times. (b) Routing overhead with varying pause times and reduced transmission power.

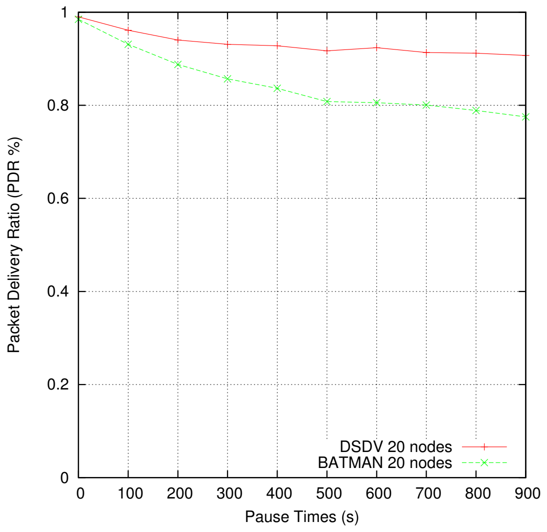
Figure 7.3: Routing overhead of BATMAN, Secure BATMAN and DSDV running with 10 nodes and 10 traffic flows.

constant load on the networks in both scenarios.

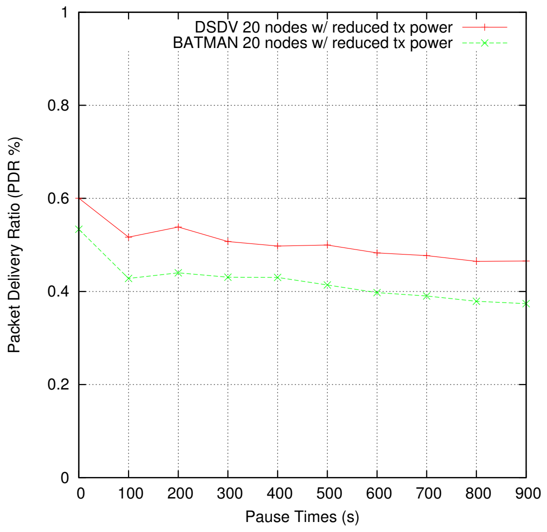
The next simulations conducted increased the number of nodes as well as the amount of traffic flows. These simulations were only conducted with the original BATMAN and DSDV due to the issues stated in Section 6.3.3.

Figure 7.4 presents the PDR results from both BATMAN and DSDV with 20 nodes both without and with reduced transmission power. The same is repeated with 30 nodes and traffic flows and also presented in Figure 7.4.

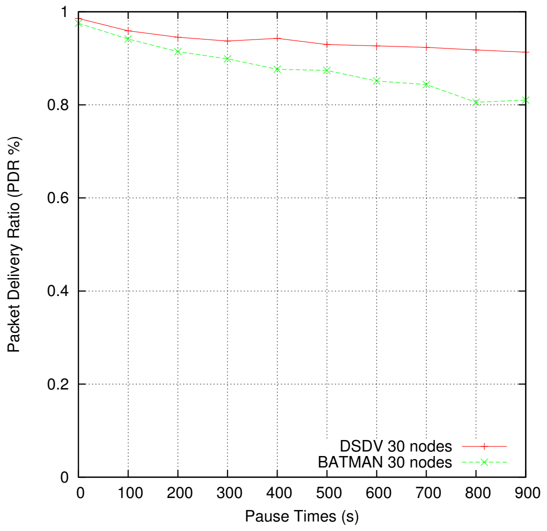
The situation depicted in the graphs in Figure 7.4, show that also here the DSDV protocol is slightly better at delivering packets than BATMAN despite the reduced transmission power.



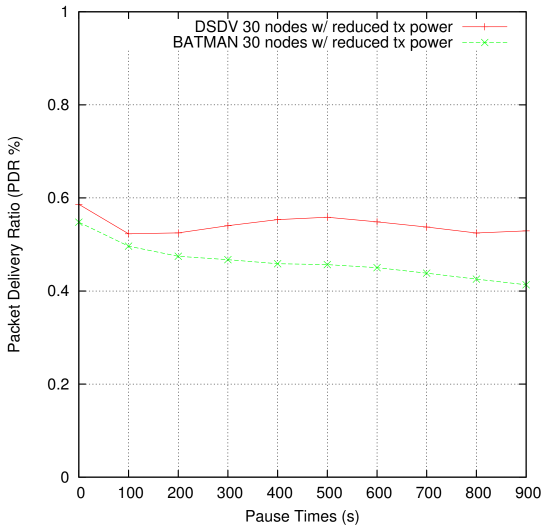
(a) PDR with 20 nodes.



(b) PDR with reduced transmission power and 20 nodes.



(c) PDR with 30 nodes.



(d) PDR with reduced transmission power and 30 nodes.

Figure 7.4: PDR values from the simulations with BATMAN and DSDV running 20 and 30 nodes and traffic flows.

Chapter 8

Discussion

This chapter discusses some of the results presented in the previous chapter pointing out important performance aspects they highlight. The chapter also discusses protocol validation and the experience of working with the ns-3 simulator. A short evaluation of the security of the Secure BATMAN protocol is also included.

8.1 Performance Results

According to the results presented in the previous chapter, the Secure BATMAN routing protocol does not perform significantly worse than its original counterpart or Destination Sequenced Distance Vector (DSDV) protocol.

The protocol's Packet Delivery Ratio (PDR) are in both scenarios, "weak network" and "strong network", at the same level as the original BATMAN and DSDV. It indicates that its ability to route and deliver application data is not significantly affected by the added security elements. The extra checks and computations performed when transmitting Authentication Messages (AMs), receiving one-time passwords, and generating key streams do not affect the total delivery rate of the protocol.

The Secure BATMAN protocol actually gives the lowest packet delay during both simulation scenarios. However, since the packet delay is measured at MAC level this entails that also routing protocols are registered. Thus the average value measured for the Secure BATMAN protocol is likely to be reduced due to the transmissions of the extra AMs as described in Section 5.8.

All three routing protocols are proactive meaning that routing information transmitted in the network will be significant as mirrored in the results. As expected, both BATMAN and the Secure BATMAN impose more load on the network compared to DSDV. However, this added routing load affects neither the protocols' PDR nor average packet delay.

The Secure BATMAN has less routing overhead than the original BATMAN. This is probably due to the periodic exchange of AMs between nodes which reduces the total amount of BATMAN packets which are flooded through the network. Thus the average routing overhead produced by the Secure BATMAN protocol is reduced.

However, despite their vastly different amounts of overhead, all three protocols have routing loads that stay nearly constant in both scenarios also with varying pause times. This is expected due to the fact they are proactive routing protocols which means that the routing information is transmitted periodically and is therefore not as affected by changes in the topology or network environment.

The original BATMAN protocol was further tested and compared against DSDV in scenarios containing more nodes and traffic sources. Simulations were done in both ”strong network” and ”weak network”. The results from these simulations show that DSDV perform slightly better than BATMAN.

It was expected that the BATMAN protocol probably would perform better or at least the same as DSDV in a network which was weaker and had lossy links. However, according to the behavior mirrored in the PDR measured during the simulations, this is not the case. BATMAN is always slightly below DSDV.

Not all aspects of the Secure BATMAN routing protocol were implemented in ns-3 as explained in Section 5.8. Also, the protocol was not tested in environments with more nodes and more traffic. However, since the protocol in the first simulations did perform similar to its original counterpart, we can assume that it would behave identically in these situations as well.

8.2 Security Design Choices of the Secure BATMAN

As Section 3.2 describes, the main goal of the security measures introduced to the BATMAN protocol, was to include a form of access control mechanism in Mobile Ad hoc Networks (MANETs) which were to be used in emergency situations and similar.

The proposed design described in Chapter 3 does present a solution which accomplishes this. It enables nodes part of a restricted network to verify if the Originator Message (OGM) received is sent from a authorized node.

However, the design does have some weaknesses and issues which should be assessed, some of which include:

- Security weaknesses
- Procedure of Proxy Certificate (PC) issuing
- Service Proxy (SP) presence
- Multiple SPs

- Network Merging

Even though the security design manages to create a network only allowing authorized nodes to send routing updates, there are some security weaknesses which can disrupt the topology of the network. Some security attacks exploiting these vulnerabilities include e.g. wormhole attack [HPJ06] and suppress replay attack [Gon92]. Both attacks are challenging to prevent in wireless networks, but can possibly be avoided by ensuring integrity and confidentiality of the routing updates (OGMs) sent. However, adding more security mechanisms must be balanced with the available resources in ad hoc networks as discussed in Section 3.1.

Some of the other issues mentioned in the list above include aspects involving the functionality of the restricted ad hoc network after the authentication mechanism has been introduced. For instance the situation where multiple SPs are present establishing several distinct restricted networks. If the different networks wish to merge, this must be solved in some matter. There is also the situation if the SP disappears, then no new nodes can join the network anymore. The security design described in Chapter 3 must be extended in order to solve these issues.

8.3 Protocol Validation

As mentioned in Section 4.3 and 5.7, the BATMAN protocol has not yet been implemented in any network simulator at the time of writing. Thus, there was no possibility of validating the behavior of the protocol implemented in ns-3 except from comparing it to the real-life protocol.

So, the behavior of the implemented protocol was verified by manually studying and comparing detailed verbose debug output from both the real-life protocol and the routing model implemented in ns-3. Seen from these tests, the protocols behave identically calculating the correct values used during routing and making the same routing decisions.

8.4 Experience Working with ns-3

ns-3 presents itself as being a powerful simulator with great capabilities and a modularity which makes it relatively easy to extend.

However, the simulator has a steep learning curve. Partly due to its higher degree of realism, the complexity of the simulator is increased compared to ns-2. Low-level details are introduced at an early stage of the implementation process increasing the time and effort required to understand the simulator well enough in order to introduce new elements. However, when this initial learning curve is conquered, the simulator shows its wide range of possibilities. In addition, the simulator's framework for data and statistics collection as well as experiment control is good.

Due to its complexity and lack of supported models, the opportunity to quickly test and study research ideas is not as trivial. In many ways ns-2 would be preferred in these situations having implemented support for more models.

ns-3 is however actively supported and developed thus the simulator should evolve and include more models and increased documentation in the future.

Running simulations is time consuming and resource demanding. It was expected to be able to run more simulations than what was actually done, but due to time and resource constraints, this was not possible. Doing several changes in between long simulation runs was not an option.

Chapter 9

Conclusion

The goal of the work behind this report was to extend the network simulator ns-3 in order for it to support the BATMAN ad hoc routing protocol and a modified version of the same protocol. The modified version, Secure BATMAN, included security elements which provided an authentication mechanism. Both implementations were then to be used to conduct simulations evaluating and comparing the protocols' performance in various scenarios.

Before the implementation in ns-3 was started, a comprehensive study was done to investigate the different network simulators available. An effort was also made to understand the background of the most prominent network simulators including their advantages and limitations.

The report presents the implementation of the BATMAN protocol in ns-3 including details about the different components, important attributes and class interactions. Due to time constraints, not all aspects of the Secure BATMAN protocol was implemented. However, the most important elements considering the affect they would have on the total routing performance, was added to the routing model in ns-3.

The main goal of the security measures introduced to the BATMAN protocol, was to include a form of access control mechanism in Mobile Ad hoc Networks (MANETs) which were to be used in e.g. emergency and rescue situations. The hope was to include these security elements without them significantly impacting the overall performance and functionality of the protocol.

The results derived from the simulations running the modified BATMAN protocol indicate that the routing performance of this protocol is not substantially affected by its security extensions.

The BATMAN protocol was also evaluated in denser networks with higher packet loss. Compared to the Destination Sequenced Distance Vector (DSDV) protocol, the results from the simulations include the fact that the BATMAN protocol does not perform as well or better in the environments in which it is supposed to be superior.

Overall it is shown that BATMAN is a protocol which can be easily extended for security purposes and the simulations indicate that the added elements do not compromise the protocols performance. Also, the original BATMAN protocol has not been implemented in any network simulator yet, thus the work done here is a good step towards it finally being supported by a network simulator.

9.1 Further Work

Future work would first of all include spending more time validating the BATMAN protocol implemented in ns-3.

Also, the remaining security elements should be added to the Secure BATMAN routing model implemented in ns-3. This includes Proxy Certificate (PC) signing and issuing by a Service Proxy (SP) and exchanging and verification of Proxy Certificate 1s (PC1s) between nodes in the network.

After the complete Secure BATMAN routing protocol is implemented in ns-3, new simulations should be conducted evaluating the impact these additional elements impose on the protocol's routing performance.

References

[BBE+99] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, et al. Improving Simulation for Network Research. Technical report, Technical Report 99-702b, University of Southern California, 1999.

[BCC+ay] Hans-Bernhard Bröker, John Campbell, Robert Cunningham, David Denholm, and more. *Gnuplot Documentation. http://www.gnuplot.info/docs_4.4/gnuplot.pdf*, Last accessed June 27, 2011. http://www.gnuplot.info/docs_4.4/gnuplot.pdf.

[BEF+02] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haibo Yu. Advances in Network Simulation. *Computer*, 2002.

[BG10] Anne Gabrielle Bowitz and Espen Grannes Graarud. Developing a Secure Ad Hoc Network Implementation. Technical report, The Norwegian University of Science and Technology (NTNU), 2010. http://github.com/annegabrielle/secure_adhoc_network_ns-3/master/ns3_source_code/raw/shared/project.pdf.

[BMJ+98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, 1998.

[BP96] Lawrence S. Brakmo and Larry L. Peterson. Experiences with network simulation. *ACM SIGMETRICS Performance Evaluation Review*, 1996.

[BRS03] Christian Bettstetter, Giovanni Resta, and Paolo Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2003.

[CBD02] Tracy Camp, Jeff Boleng, and Vanessa Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless communications and mobile computing*, 2002.

[CJay] Thomas Heide Clausen and Philippe Jacquet. Optimized Link State Routing Protocol (OLSR). *Network Working Group*, Last accessed June 27, 2011. http://tools.ietf.org/html/rfc3626.

[Dir07] National Police Directorate. *Police Emergency Prepareness System, Part I Norway - Emergency Manual*. someone, 2007.

[DPR00] Samir R. Das, Charles E. Perkins, and Elizabeth M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. IEEE, 2000.

[FKGay] Sheila Frankel, Scott Kelly, and Rob Glenn. The AES-CBC Cipher Algorithm and Its Use with IPsec. *Network Working Group*, Last accessed June 27, 2011. http://www.faqs.org/rfcs/rfc3602.html.

[Fri46] Harald T. Friis. A note on a simple transmission formula. *Proceedings of the IRE*, 1946.

[GKS95] Nada Golmie, Alfred Koening, and David Su. *The NIST ATM network simulator: Operation and programming*. National Institute of Standards and Technology, 1995.

[Gon92] Li Gong. A security risk of depending on synchronized clocks. *ACM SIGOPS Operating Systems Review*, 1992.

[Gra11] Espen Grannes Graarud. Implementing a Secure Ad Hoc Network. Technical report, The Norwegian University of Science and Technology (NTNU), 2011. http://github.com/espengra/secure-ad-hoc-network-doc/raw/master/share/thesis.pdf.

[HPFS02] Russell Housley, Tim Polk, Warwick Ford, and David Solo. Internet x. 509 public key infrastructure certificate and certification revocation list (crl) profile. *Network Working Group*, 2002. http://tools.ietf.org/html/rfc3280.

[HPJ06] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 2006.

[HRFR06] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. ns-3 Project Goals. In *Proceeding from the 2006 Workshop on ns-2: The IP Network Simulator*, 2006.

[HRFRay] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. *Developing the Next-Generation Open-Source Network Simulator (ns-3). http://www.nsnam.org/docs/meetings/snowbird06/ns-3-cri-workshop-2006.pdf*, Last accessed June 27, 2011. http://www.nsnam.org/docs/meetings/snowbird06/ns-3-cri-workshop-2006.pdf.

[Kesay] Srinivasan Keshav. *REAL 5.0 Overview. http://www.cs.cornell.edu/skeshav/real/overview.html*, Last accessed June 27, 2011. http://www.cs.cornell.edu/skeshav/real/overview.html.

[Mes11] Open Mesh. *BATMAN Documentation. http://gitorious.org/batman-adv-doc*, Last accessed May 4, 2011. http://gitorious.org/batman-adv-doc.

[Mesaya] Open Mesh. *FAQ. open-mesh.org*, Last accessed June 27, 2011. http://www.open-mesh.org/wiki/open-mesh/FAQ.

[Mesayb] Open Mesh. *The OLSR Story. open-mesh.org*, Last accessed June 27, 2011. http://www.open-mesh.org/wiki/the-olsr-story.

[NAL07] Axel Neumann, Corinna Aichele, and Marek Lindner. B.A.T.M.A.N Status Report. Technical report, 2007. http://downloads.open-mesh.org/batman/papers/batman-status.pdf.

[NALWay] Alex Neumann, Corinna Aichele, Marek Lindner, and Simon Wunderlich. Better Approach To Ad-Hoc Networking (B.A.T.M.A.N) draft-wunderlich-open-mesh-manet-routing-00. *Network Working Group*, Last accessed June 27, 2011. http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00.

[NCÇ+11] Hemanth Narra, Yufei Cheng, Egemen K. Çetinkaya, Justin P. Rohrer, and James P.G. Sterbenz. Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3. *something*, 2011.

[nsnaya] nsnam. *ns-3 Model Library. http://www.nsnam.org/*, Last accessed June 27, 2011. http://www.nsnam.org/docs/release/3.11/models/ns-3-model-library.pdf.

[nsnayb] nsnam. *ns-3 Tutorials and Manual. http://www.nsnam.org/tutorials.html*, Last accessed June 27, 2011. http://www.nsnam.org/tutorials.html.

[nsnayc] nsnam. *Official ns-2 Website. http://www.isi.edu/nsnam/ns/*, Last accessed June 27, 2011. http://www.isi.edu/nsnam/ns/.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 1994.

[PBRD03] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. *Network Working Group*, 2003.

[RSA78] Ronald Linn Rivest, Adi Shamir, and Leonard Max Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[SQLay] SQLite. *About SQLite. http://www.sqlite.org/about.html*, Last accessed June 27, 2011. http://www.sqlite.org/about.html.

[TET+ay] Steven Tuecke, Doug Engert, Mary Thompson, Laura Pearlman, and Von Welch. Internet X.509 Public Key Infrastructure Proxy Certificate Profile. *Network Working Group*, Last accessed June 27, 2011. http://tools.ietf.org/html/rfc3820.

[TJÅAN09] Inger Anne T∅ndel, Martin Gilje Jaatun, and Åsmund Ahlmann Nyre. Security requirements for manets used in emergency and rescue operations. In *Security and Communication Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on*, 2009.

[Ver11] Amandeep Verma. A study of performance comparisons of simulated ad hoc network routing protocols. 2011.

[VMC02] John Viega, Matt Messier, and Pravir Chandra. *Network Security with OpenSSL*. O'Reilly Media, 2002.

[WvLW09] Elias Weingartner, Hendrik vom Lehn, and Klaus Wehrle. A performance comparison of recent network simulators. In *Communications, 2009. ICC '09. IEEE International Conference on*, 2009.

[YLY+04] Hao Yang, Haiyun Luo, Fan Ye, Songwu Lu, and Lixia Zhang. Security in mobile ad hoc networks: Challenges and solutions. *Wireless Communications, IEEE*, 2004.

Appendix A

BATMAN Routing Protocol Details

A.1 Host Network Announcement (HNA)

A HNA message has a fixed size of 5 Bytes. A node appends one or more HNAs when it wants to announce a gateway to another network or host.

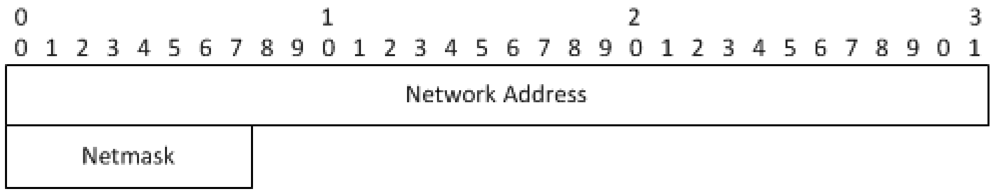


Figure A.1: HNA message Format.

The different fields found in this message are:

- **Netmask**
  Indicates the size of the announced network.
- **Network Address**
  The IPv4 network address of the announced network.

A.2 Sequence Numbers and Sliding Window

Sequence numbers cycle from 0 to $2^{16} - 1$ and start from 0 again when reaching the maximum value. Since the number range is limited, all arithmetical operations done must be performed using using modulo 2^{16}.

In-Window Sequence Numbers:
These numbers represent the latest accepted sequence numbers. The window ranges from the current sequence number of the originator to `WINDOW_SIZE` - 1 sequence numbers below it. The current sequence number of a node is

not updated if an Originator Message (OGM) from this originator is received containing an In-Window Sequence Number. It is however marked in the sliding window that an OGM with this In-Window Sequence Number has been received.

Out-Of-Range Sequence Numbers:
These are all the sequence numbers that are not within the In-Window range and are considered as the the new or next-expected Sequence Numbers. The current sequence number is set to the sequence number in an received OGM if this number is out of range. Thus the sliding window is moved and the number which are not inside the window anymore are dropped.

A.3 Neighbor Ranking in BATMAN IV

The BATMAN IV needs to keep track of two different Transmit Link Quality (TQ) values:

- The local TQ value which is the transmission quality towards every link-local neighbor
- The global TQ value which is the link quality towards every multi-hop neighbor

The local TQ towards a link-local neighbor is calculated using the Receiving Link Quality (RQ) and Echo Link Quality (EQ) value of the neighbor. The global TQ value is the average of all recently received TQ values in OGMs received from a distinct neighbor. This global TQ is now used when choosing the best neighbor as next-hop towards a neighbor.

A.4 Sliding Windows in BATMAN IV

The BATMAN algorithm keeps track of received packets over a time interval using a sliding window similar to the one explained in Section A.2. The amount of packets (sequence numbers) registered in a neighbors sliding window, is referred to as the neighbor's RQ value.

The BATMAN also keeps a sliding window to keep track of the received TQ values in OGMs which is used to calculate the global TQ used in neighbor ranking.

Appendix B

Hardware Details

The simulations were performed on a computer with the following hardware specifications:

- Intel Core 2 Duo 2.83 GHz processor
- 4 GB memory

In addition, the computer was running Ubuntu 10.4 LTS - the Lucid Lynx (Linux Kernel 2.6.32-25-generic-pae).

Appendix C

Simulation Script

C.1 Script

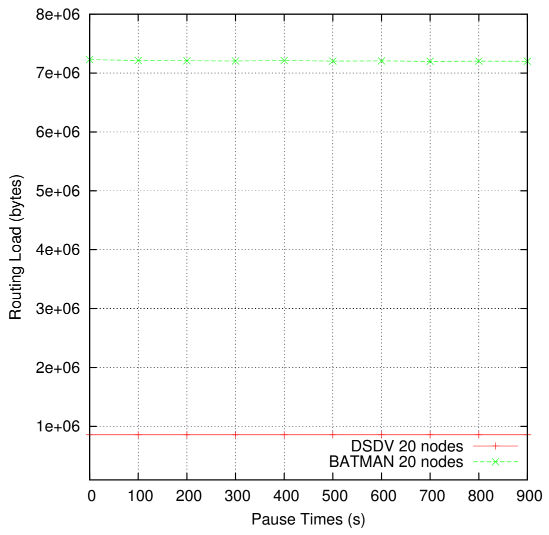
```
#!/bin/sh

PAUSE="0 100 200 300 400 500 600 700 800 900"
TRIALS="1 2 3 4 5 6 7 8 9 10"

echo WiFi Experiment Example

pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example-sim does not)
      ."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example-sim does not)
      ."
  exit 255
fi

pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim does not)."
  exit 255
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

# Remove existing database
if [ -e data.db ]
then
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm data.db
  fi
fi

# Compile the simulation scenario with the different parameters, and
   run the number of trials
for pause in $PAUSE
do
  for trial in $TRIALS
  do
    echo
    echo Pause time $pause, Trial $trial
    export NS_GLOBAL_VALUE="RngRun=$trial"
    ./waf --run "main-scenario --format=db --batman=1 --pause=$pause --
        start=30 --stop=930 --sources=20"
  done
done

# Create SQL command to get packet delivery ratio
PDR_CMD="SELECT rx.run, avg(cast(rx.value as real)/cast(tx.value as
    real))
    FROM Singletons rx, Singletons tx
    WHERE rx.variable = 'onoffRx' AND tx.variable='onoffTx'
    GROUP BY rx.run
    ORDER BY rx.run ASC;"

# Get OnOff packet delay results
DELAY_CMD="SELECT onoff.run, (avg(onoff.value)/1000000000)
  FROM Singletons onoff
  WHERE onoff.variable = 'onoffDelay-average'
  GROUP BY onoff.run;"

# Get MAC packet delay results
MAC_CMD="SELECT mac.run, (avg(mac.value)/1000000000)
  FROM Singletons mac
  WHERE mac.variable = 'macDelay-average'
  GROUP BY mac.run;"

# Amount transmitted and received BATMAN packets AVG!
BATMAN_CMD="SELECT batmant.run, batmant.value, batmanr.value
  FROM Singletons batmant, Singletons batmanr
  WHERE batmant.variable = 'batmanTx' AND batmanr.variable = 'batmanRx'
  GROUP BY batmant.run;"

# Amount transmitted and received DSDV packets AVG!
DSDV_CMD="SELECT dsdvt.run, dsdvt.value, dsdvr.value
  FROM Singletons dsdvt, Singletons dsdvr
  WHERE dsdvt.variable = 'dsdvTx' AND dsdvr.variable = 'dsdvRx'
  GROUP BY dsdvt.run;"
```

```
# Get everything!
ALL_CMD="SELECT * FROM Singletons;"

# Query the SQLite Database
sqlite3 -noheader data.db "$PDR_CMD" > test.data

# Parse the data
sed -i "s/run-//" test.data
sed -i "s/|/    /" test.data

# Run gnuplot script and create graph
#gnuplot main-delay.gnuplot
#gnuplot main-overhead.gnuplot
gnuplot main-pdr.gnuplot

echo "Done;"
```

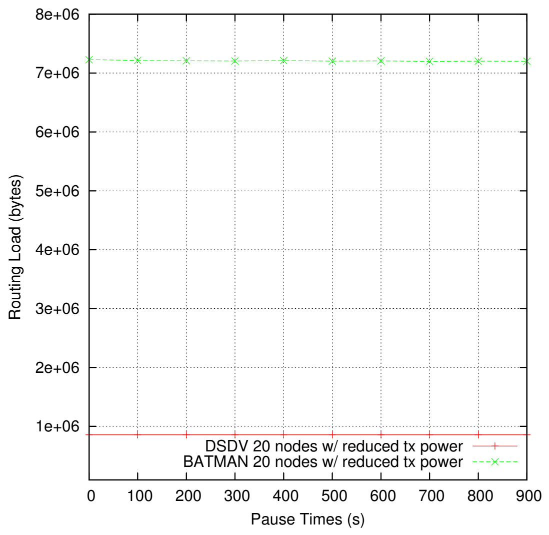
Appendix D

Additional Simulation Results

Figure D.1 show the routing overhead produced by the original BATMAN protocol and the DSDV protocol running 20 nodes and traffic flows in two scenarios.



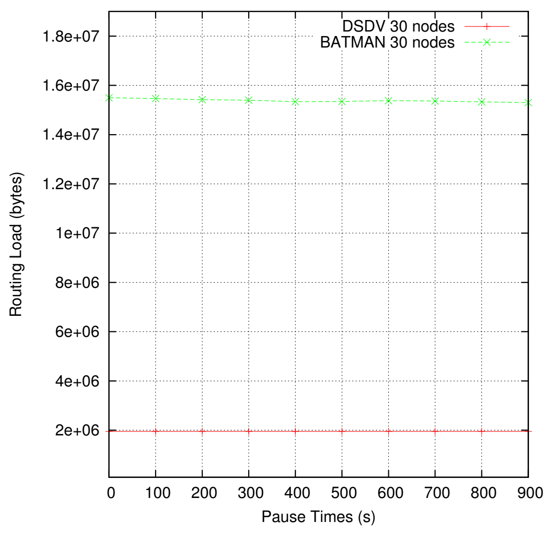
(a) Routing overhead with varying pause times.



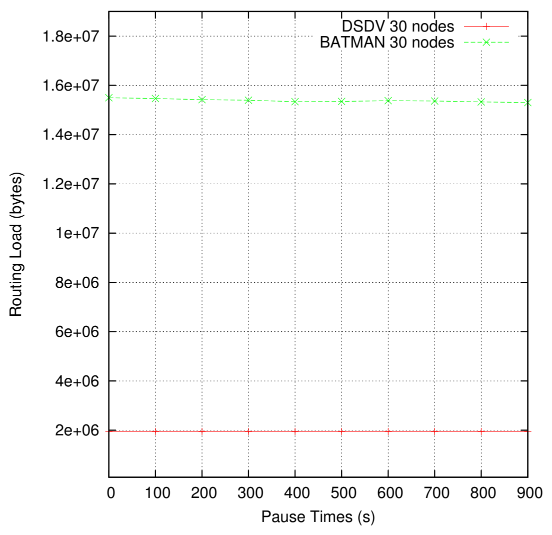
(b) Routing overhead with varying pause times and reduced transmission power.

Figure D.1: Routing overhead produced by BATMAN and DSDV running with 20 nodes and 20 source and sink pairs where the transmission power has been reduced.

Figure D.2 show the routing overhead produced by the original BATMAN protocol and the DSDV protocol running 30 nodes and traffic flows in same scenarios as above.



(a) Routing overhead with varying pause times.



(b) Routing overhead with varying pause times and reduced transmission power.

Figure D.2: Routing overhead produced by BATMAN and DSDV running with 30 nodes and 30 source and sink pairs where the transmission power has been reduced.

Appendix E

Paper

The following paper was written based on this master thesis as well as the work done by Espen Grannes Graarud in his master thesis [Gra11]. It was written together with Martin Gilje Jaatun, Dr. Lawrie Brown, and Espen Grannes Graarud and has been submitted to the ICDIM 2011 Conference [1].

[1]http://www.icdim.org/

BatCave: Adding Security to the BATMAN Protocol

Blind review

Abstract—**The Better Approach To Mobile Ad-hoc Networking (BATMAN) protocol is intended as a replacement for protocols such as OLSR, but just like most such efforts, BATMAN has no built-in security features. In this paper we describe security extensions to BATMAN that control network participation and prevent unauthorized nodes from influencing network routing.**

I. Introduction

This work developed from a perceived need to implement a secure adhoc network that might be used in emergency services, disaster assistance, and military applications. Such a network needs to be established quickly, and without the need for existing fixed infrastructure. However it also requires controls to limit access to the network, in order to protect it from intruders or unwanted bystanders. We propose extensions to a suitable adhoc network routing protocol, BATMAN, so that routing advertisements will only be accepted from authorised stations on the network. We propose the use of proxy certificates, which each client wishing to access the network will generate, and which are signed by one of the suitably authorised stations tasked with creating and managing the network. We assume these stations will be located with suitable emergency services command units that the network is being created to support.

The remainder of this paper is structured as follows:

II. Related Work on Adhoc Network Security

Our proposals evolved from work on developing a secure restricted ad-hoc network for use by emergency services or disaster response personnel [1], [2]. In such a network, access must be managed, but be provided for members of multiple authorities which might not have online access to verify their identity. They focused on the design and implementation of the needed extensions to the OLSR adhoc network routing protocol. However they only made a brief mention of the use of a public-key infrastructure to identify mobile clients and to authorise their access to some restricted ad-hoc network. They suggested that clients in a region would be pre-configured with certificates that could be used to automatically grant them access. They also noted that there needs to be some means of granting access to mobile devices that are not known, for personnal from out of region or from other services without peering arrangements. They suggested that such devices can be issued short-lived certificates, with limited rights, to grant them access. However details of this were left mostly unspecified.

In other related work, short-lived X.509 certificates were proposed as a suitable mobile authentication method for low power or otherwise resource limited devices [3], [4]. The main reasons they gave for choosing such certificates, which are "conventional" X.509 certificates but with a much shorter lifetime of hours to days, include a desire to avoid the cost and overhead of checking a Certificate Revocation List (CRL) or otherwise handling detection of revoked certificates. It was also to allow the use of less computationally intensive algorithms and key sizes than may be required in "conventional" X.509 certificates with lifetimes, and hence need for sufficient strength against attack, over periods of months to years.

III. Addressing Limitiations in the Existing Work

Our proposed adhoc network security extensions address some issues with the prior work noted above. First was the choice of adhoc network routing protocol to modify. Although OLSR is an Internet standard, several papers have suggested thats its performance in practical trials is less than desired [5], [6]. Of the other protocols tested, it appears that BATMAN provided the best overall performance. We present further details on this choice in the next section.

Next was the choice of types of certificates to use to manage controlled admission to the network. The existing proposals involve using a mix of conventional and short-lived certificates, with the latter being generated in the field as required to support admission of stations without existing, verifiable, conventional certificates. However this means the stations issuing these need to support some certificate authority (CA) functionality, and have CA certificates available to sign these newly created certificates (short-lived or otherwise). Normal client stations would not normally have these.

We propose instead the use of proxy certificates, which are X.509 certificates with specific proxy extensions, that are signed either by another, conventional client certificate, or by a proxy certificate (PC), as we detail later in section VI-A. Hence any client station can potentially act a certificate issuer, able to grant access to other stations. The problem then becomes one of distributing knowledge of which stations have that authority, which we address as part of our protocol extensions. Note with our proposed use of proxy certificates, they become an access token or capability used to gain access to a service, in this case the adhoc network. This is very much the opposite sense to current use of these certificates, which are used by clients to delegate some of their access rights to a server, particularly in the grid computing domain [7].

Another problem not explicitly addressed in the previous work, is just what controls or restrictions were placed on the process of issuing certificates to grant access to the network. They identify the need to support differing categories of stations needing access. Some may be automatically recognized and trusted because they possess a conventional client certificate issued by a CA known to the proxy issuing client, most likely because both stations belong to the same service or administrative structure. In this case it would be reasonable to automatically issue the proxy certificate and grant network access without any human intervention. Other clients may not be immediately recognized, since they belong to other services, are volunteers, or just not previous known. In such cases it would seem reasonable to require manual verification that the client should be granted access before issuing a proxy certificate to them.

A further advantage in the use of proxy certificates is that they support the specification of restrictions on their use. We propose using this mechanism to assign different rights to different classes of clients. This could be used to indicate which clients are delegated the right to also issue proxy certificates granting access to other stations to the existing network. It also could be used to indicate that some stations should only be end-systems, and not used to relay traffic. Since X.509 certificates are widely recognized, it would also be possible to use the issued proxy certificates to authorise and authenticate the client's use of specific upper-layer applications.

IV. B.A.T.M.A.N.

BATMAN [8] ("Better Approach To Mobile Ad hoc Networking") is an increasingly popular routing protocol for wireless ad hoc networks, which was developed with an aim to replace the Optimized Link State Routing Protocol (OLSR) [9]. OLSR is a pro-active routing protocol, which means that participating nodes regularly exchange routing information with each other. According to the BATMAN developers, the problem with OLSR is that every node in the network calculates the whole routing path, which is a complex way to do it. Not only is it difficult to make sure all nodes have the same information at the same time, it also needs (relatively) much storage and computation time. If nodes sit on different routing information this concept leads to routing loops and heavy route flapping. The result is many patches to the protocol that defies the protocol standard in order to make it more suitable [9].

In BATMAN, each node should only know the next hop, i.e., the link-local neighbor that is the path between itself and the destination. BATMAN calculates the optimal route, i.e. the next jump, by comparing the number of routing messages it has received from each node and who was the last sender.

The routing messages sent in BATMAN are called OGM. Figure 1 shows the packet format with all header fields. The OGM format has changed since the BATMAN draft [8] was published, but there is no official publication with the new packet format as of yet. The packet format found in the RFC draft belongs to the older version III of the BATMAN algorithm. The algorithm used in this paper is version IV.

The real workhorse of the packet is the "Originator Address" field which carries a host address of the node 'A' that broadcasted the OGM. When a node 'B' receives this message it checks if the originator address and source address of the IP header are the same - if so the two nodes are direct neighbors. B then forwards the OGM only changing the "TTL" and "Previous Sender" fields. All OGM inside the BATMAN network are broadcasted and rebroadcasted until the TTL has dropped to zero, or until they receive an OGM they have previously sent themselves.

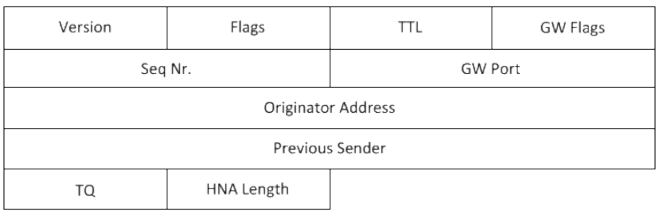


Fig. 1: BATMAN's OGM packet format.

This way all OGM will be received and rebroadcasted by all nodes in the network and all nodes will learn the existence of each other and which nodes are the first hop between them and the other nodes, i.e. the first leg of the path. All nodes and their first hops in their paths are stored in a list called an "Originator List".

When a node which has already received and forwarded an OGM receives the same OGM from another node at a later point - it drops that packet so the network will not get flooded by forwarding the same OGM until its TTL is zero. This is also necessary in order to prevent routing loops.

V. Requirements

Ad hoc networks have some desired characteristics such as quick and inexpensive setup and being independent of communication infrastructure, but they also introduce great challenges regarding security.

A. Scenario

The design and implementation presented in this paper is mostly based on an emergency situation scenario, in which communication infrastructure is unavailable. If there is a major emergency situation such as an earthquake or tsunami, it is likely that parts or the entire communication infrastructure at the scene is destroyed or temporarily down. The remaining communication lines will then probably be congested, such that little communication actually goes through.

In this situation, it is of great importance that Emergency Personnel, such as Paramedics, Firemen, Policemen and the Military, are able to communicate efficiently and therefore independently of the public communication infrastructure. They need this network in order to manage the the operation, and therefore availability is probably the most important trait of this network. Secondly, they should be able to trust the communication on the network – i.e., messages sent are from whom they claim they to be.

Also, being able to authorize new actors on the scene, such as Red Cross, can be critical to the operation. These new actors will probably not have the necessary authentication tokens, i.e. certificates, required by the authentication scheme in the network.

B. *List of Requirements*

Based on the scenario above these requirements can be extracted and made into general requirements that needs to be addressed by the system design. The work presented here is based on several sources, most prevalent being the research from the OASIS project [2] [10] [1] and Winjum et al. [11].

R1	A node must be authorized in order to get full rights in a network [12], [13]
R2	A node without a recognized authentication token should be able to become authorized if necessary
R3	Networks need a master node which handles access control
R4	Access control (after initial authentication) should not rely on centralized nodes
R5	Different networks should be able to collaborate [11]
R6	Only master nodes can decide access policies of users/nodes
R7	Nodes must not be able to alter access policies they are ruled by

An early study produced security requirements of ad hoc networks demanding that the routing logic must not be spoofed or altered to produce different behavior [12]. This means authorization is required (R1) before someone can partake in routing logic. The OASIS project [2] specifically considered a situation where e.g. NGOs contribute to a rescue operation, which means they need to somehow acquire credentials (R2), but this must be administered by some authority (R3). R4 highlights the need for authenticated nodes to function autonomously. A desire for seamless radio coverage over the area gives us R5. R6 comes from the fact that it is not possble to determine access policies prior to network setup, and R7 states the rather obvious, in that nodes that could alter the access policy would violate R6.

VI. Security Solution Overview

The system design requires nodes to be authenticated and trusted before being allowed into the network. Each node also has to verify their identity periodically, or they are dropped from the network.

The network setup starts with an out-of-band authentication where a master node, hereafter referred to as a Service Proxy (SP), verifies new nodes. How this is done can be up to the application, but let us assume that the actors carrying their communication devices, hereafter nodes, physically meets the SP at the scene and exchange their public key fingerprints.

When a new node is discovered by the SP using regular routing announcements as part of the pro-active routing protocol, the SP will invite the new node to a handshake to establish a trust between the two nodes. The new node will receive the SP's certificate, and will after verifying the fingerprint request a proxy certificate for itself. After verifying the node's fingerprint, the SP will issue a proxy certificate with (possibly) the rights to participate in building the MANET by broadcasting its own and re-broadcasting other trusted nodes' routing announcements.

A. *Why use Proxy Certificates?*

. The Proxy Certificate (PC) is used to delegate rights on behalf of the issuer. That means that the issuer, i.e. the SP, can choose to delegate all or a subset of its rights to the receiver of the Proxy Certificate. This can be very useful in a situation where the nodes themselves are unable to properly authenticate themselves with their pre-existing conventional X.509 certificate if the SP on the scene has no way to verify their certificates. This can be true if their certificates are issued by an unknown root certificate (CA) or simply if there is no Internet access and the certificate is signed by an unknown entity (unknown to the SP), even if it knows and trusts the root CA.

Also, the SP could be interested in giving the node rights the node would not usually have on this specific scene, depending on the situation. This is easier to achieve when the SP can delegate its own rights.

An important feature of the PC is that the SP can delegate different kind of rights, as long as it is a subset of its own rights, to different nodes. There are countless of different rights that can be useful, given the situation they are used in, but here is a few possible rights/privileges to give the reader an understanding of the possibilities they give:

- Announce itself - let the MANET know of your existence
- Re-broadcast other nodes announcements - reshape the network topology
- Announce a gateway - give the MANET access to another network
- Use the gateway - allow you to communicate outside the MANET
- Send and receive messages with a defined application - full application rights
- Only receive messages from a defined application - limited application rights

If you are setting up a MANET on the scene of a disaster to assist emergency personnel, you could have some actors be able to organize the effort by sending orders/commands to the other actors, while some actors only are allowed to receive the orders. In this situation it might be of great importance to know that only verified nodes are able to give commands, but the importance of getting this information available outweighs the need to verify the nodes/actors receiving this information.

B. *Post-Authentication Operation*

After being issued with a Proxy Certificate (PC) the newly authenticated node will periodically "broadcast" - unicast to each neighbor - a message containing an ephemeral key and corresponding Initialization Vector (IV), a pseudo-randomly generated nonce, and a digital signature over this message. The ephemeral key is encrypted with the neighbor's public key (hence multiple unicasts instead of an actual broadcast), but the digital signature is generated based on the unencrypted key and the other contents of the message, and is thus identical for all neighbors.

After sending this signed "broadcast" to each neighbor, the node and its neighbors will generate a keystream from the ephemeral key, IV, and nonce. The node will then append two new bytes from this keystream to each routing announcement, and re-broadcasts of neighbors' announcements, sent from this point forward with a sequence number for the recipient to be able to match this "extract" with the keystream at an offset given by the sequence number. The two bytes will then in effect be a one-time password similar to that used by some online banking applications. If this one-time password value is absent or incorrect, the announcement will be dropped and regarded as a spoofing message.

Whenever a routing announcement is re-broadcasted by another trusted node, that node will first replace the sequence number and one-time password that it has verified with the next two bytes of its own key stream. This means that every node only checks its direct neighbor for authentication, which is a design choice. This proposal assumes that because every node is verified by the SP in the first place, all nodes in the network will be able to trust each other, which also means they will trust their neighbors to properly verify their neighbors again.

In order for trusted nodes to learn of newly trusted nodes existence, the SP regularly broadcasts lists containing the id, address and public key of each trusted node in the network. This needs to be done, because before learning about a new node the other trusted nodes will not accept any messages from this node. This means the new node will not be able to exchange its own PC with other nodes directly - only through the SP.

The list, hereafter Authentication List (AL), also adds some web-of-trust like capabilities. The list is signed by the SP, which means the integrity of the list is guaranteed by the SP. This means that if the SP should go offline, e.g. it could be out of range, other trusted nodes in the MANET can continue to broadcast the AL on behalf of the SP - to ensure all nodes in the network know each other. This can be especially important when the network grows large and become fully or partially separated and nodes in one part may not have learnt of the existence of newly trusted nodes yet. It also applies to trusted nodes who have been offline while new nodes have been verified, then re-enter the network while the SP is offline.

VII. Simulations

We have implemented both standard BATMAN and the version with our security enhancements in the network simulation package ns3.

Figure 2 presents Packet Delivery Ratio (PDR) and packet delay results from the simulations running Secure BATMAN, BATMAN and DSDV with 10 nodes and 10 traffic flows.

As seen from Figure 2a, the PDR values of all three routing protocols all well above 80%. Interestingly, Secure BATMAN's PDR values also stay at approximately the same level as the two other protocols. At pause time zero, which is equivalent to continuous node movement, all three protocols show their best behavior with the highest PDR values. This is probably due to the fact that they all are ad hoc network protocol tailored for networks with high node mobility.

When looking at the end-to-end latency in Figure 2b it is surprisingly the Secure BATMAN protocol which has the best results.

VIII. Prototype

We have implemented our proposed protocol changes by modifying the BATMAN code distributed with a recent Ubuntu Linux distribution.

A. Initialization Phase

Figure 3 presents neighbor discovery results for both the original (Fig. 3a) and modified (Fig 3b) version of BATMAN. The two graphs shows the time in seconds on the y-axis and the trial/run number on the x-axis. The two colored lines on the graphs show the results from first neighbor discovery until the first neighbor is added to routing table (green line - marked with "x") and until both nodes are added to the routing table (red line - marked with "+").

The results from the original protocol, shown in Figure 3a, shows high variance in the time needed to add one and two nodes to the routing table. For 7 out of 10 "first nodes" the time needed is relatively equal, being about one second. For both nodes to be added however, there are much more variance - variying from the best possible time, i.e. equal to adding one node, and up above 3 times longer than adding one node. '

Figure 3b shows the results from the modified version proposed in this thesis. These results indicate that the behaviour of the modified version seems to correlate with the behaviour expected from the hypothesis. A seemingly constant of about two seconds seems to be added to the process of adding both nodes to the routing table.

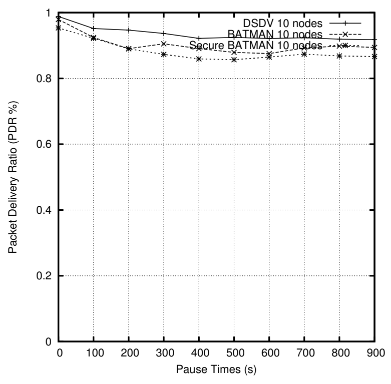
Another interesting observation is that the time variance seems to be much less than that of the original version. This might be because the authentication handshake and the keystream sharing happens in a separate thread from the regular BATMAN operations, meaning the BATMAN protocol continously receives routing announcements to process while the Authentication Module (AM) handles its part. The idea being that while the AM thread runs the BATMAN thread "gets ready" to do its part of the job.

B. Route Convergence

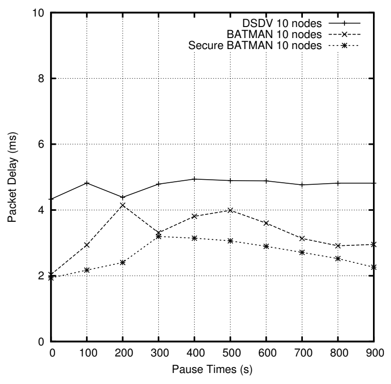
The results of the second test are shown in Figure 3c. In this figure, the axes are the same as in the figures above: y-axis shows the time in seconds, and the x-axis shows the trial run. The red line shows the performance of the original implementation, while the green line shows the modified.

As indicated earlier, this test's results are somewhat unclear. While the results using the original implementation seems relatively uniform, with only about 1 second variance, the results from the modified implementation are highly irregular.

Looking through the logs from this test one thing become apparent. With different hardware on the different nodes in the network, their wireless cards send at different levels of transmission power, meaning that while one node can receive

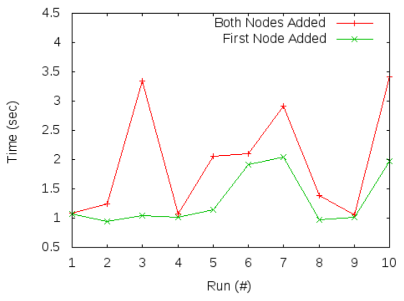


(a) PDR with varying pause times.

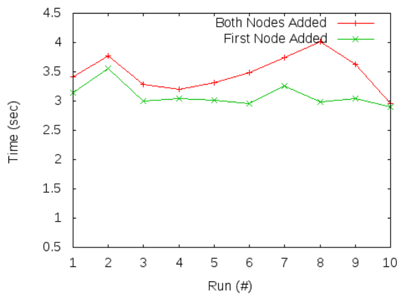


(b) Packet delay with varying pause times.

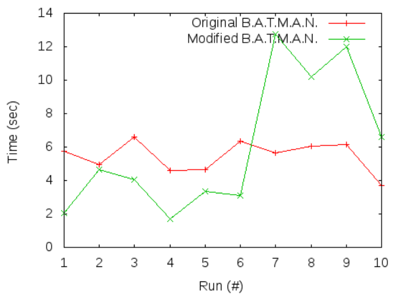
Fig. 2: Simulations results from BATMAN, Secure BATMAN and DSDV (10 nodes and 10 source and sink pairs)



(a) Original BATMAN



(b) Modified BATMAN



(c) Routing path convergence time

Fig. 3: Neighbor discovery for original and secure BATMAN, and routing path convergence

packets from a "stronger node", the packets sent might not be received by the other nodes.

The BATMAN protocol messages (routing announcements) are sent quite often, depending on the number of re-broadcasts being sent, meaning the time from when a node is within transmitting range and until its broadcasts are received by nodes within its transmitting range will be quite short. The AM messages however, was mostly tested in an ideal environment where most packets were received, so this was not properly accounted for. Therefore, if a routing announcement from a "stronger node" is received by a "weaker node", the weaker node might send its keystream material without the other node receiving it.

Re-transmitting mechanisms based on guessing that the receiving node has not received the AM messages are in place, but as the mechanism wait until it beleives the other node has not received, instead of knowing it instantly. This can of course be managed adding ACK'ing to each AM message, which was not added initially because of the wish to minimize overhead. This however, might have to be re-evaluated.

Another thing to notice is how multiple trial runs using the modified version actually performed better than the original version. This is impossible to explain talking about the design and implementations themselves, but is probably most accuratly explained in the terms of external environment.

IX. Discussion

The proposed system design uses a novel solution to continuously verify routing announcements received from one's neighbors.For this system to be used on typical mobile devices with all their constraints, limitations on computing power, battery lifetime, and saturation in the wireless network must be acknowledged.

Because all nodes in a MANET using a pro-active routing protocol broadcast their routing announcements and forward all received routing announcements, the network traffic will increase exponentially to the amount of nodes in the network and how closely bound they are. Therefore all routing announcements need to be as small as possible. A typical signature is usually one or two orders of magnitude larger than a regular routing announcement, so by adding a signature to the routing announcement - most of the data sent in the network would be signature data. This is far from ideal.

The first solution that one would think of would be to only sign a very few of the announcements, periodically. This however, would be totally disastrous. This would have no protection against spoofing attacks whatsoever, as an attacker could wait for a legitimate node to send a signed announcement and then send his own fake announcements spoofed with the legitimate node's address.

The solution proposed in this paper solves the problem in a different manner. Since each node and its neighbors generate a key stream that can be used to verify messages from that node, only messages with a correct, previously unused, "one-time password" will be accepted and forwarded by any neighbor. Furthermore, since the keystream has to be renewed periodically, any node not possessing the correct proxy certificate will be dropped from the network upon renewal.

This scheme is fully based on trust. You trust that your trusted nodes will only send you its own annoucement (correctly) and rebroadcast only its trusted nodes announcements without modification. If for some reason a trusted node should behave maliciously, this scheme will not detect this and allow the trusted node to potentially disrupt the network.

X. Conclusion

We have presented a security extension to the BATMAN ad hoc routing protocol which handles controlled network admission and prevent unauthorized nodes from influencing routing decisions in the network. Our ns-3 simulations indicate that the security mechanisms do not place an undue burden on the network nodes, and our protoype implementation confirms that although further refinements are desirable, BatCave represents a viable securty solution for ad hoc networks.

References

[1] A. Nyre, M. Jaatun, and I. Tøndel, "A secure MANET routing protocol for first responders," in *Security and Communication Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on*. IEEE, 2009.

[2] I. S. Svagård (editor), "Information security for field workers in crisis situations," SINTEF ICT, http://www.oasis-fp6.org/documents/OASIS_SP24_DDD_253_security_SIN_1_0_pub.pdf, Tech. Rep., 2008.

[3] P. K. Sharma, "Short-Lived Certificates as a Mobile Authentication Method," MSc Thesis, 2009. [Online]. Available: http://orbit.dtu.dk/getResource?recordId=245323&objectId=1&versionId=1

[4] M. Pitkanen and H. Mikkonen, "Initalizing mobile user's identity from federated security infrastructure," in *Proceedings of the Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 08)*, 2008, pp. 390–394. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/UBICOMM.2008.64

[5] M. Reineri, C. Casetti, and C.-F. Chiasserini, "Routing protocols for mesh networks with mobility support," in *Proceedings of the 6th international conference on Symposium on Wireless Communication Systems*, 2009, pp. 71–75. [Online]. Available: http://ieeexplore.ieee.org/iel5/5277434/5285213/05285344.pdf?arnumber=5285344

[6] M. Abolhasan, B. Hagelstein, and J. C.-P. Wang, "Real-world performance of current proactive multi-hop mesh protocols," in *15th Asia-Pacific Conference on Communications (APCC09)*, Oct. 2009, pp. 44–47. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5375690

[7] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist, "X.509 Proxy Certificates for Dynamic Delegation," in *Proceedings of the 3rd Annual PKI R&D Workshop, Gaithersburg MD, USA*, 2004.

[8] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better Approach To Ad-Hoc Networking (B.A.T.M.A.N) draft-wunderlich-open-mesh-manet-routing-00," *Network Working Group*, Last accessed December 19, 2010, http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00.

[9] O. Mesh, "*Why starting B.A.T.M.A.N.?*" *open-mesh.org*, Last accessed december 19, 2010, http://www.open-mesh.org/wiki/why-starting-batman.

[10] I. Tøndel, M. Jaatun, and A. Nyre, "Security requirements for MANETs used in emergency and rescue operations," in *Security and Communication Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on*. IEEE, 2009.

[11] E. Winjum, P. Spilling, and Ø. Kure, "Ad Hoc networks used in emergency networks : the Trust Metric Routing approach," FFI Rapport, Tech. Rep., 2006.

[12] B. Dahill, B. Levine, E. Royer, and C. Shields, "A secure routing protocol for ad hoc networks," *Electrical Engineering and Computer Science, University of Michigan, Tech. Rep. UM-CS-2001-037*, 2001.

[13] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "A secure routing protocol for ad hoc networks," *Network Protocols, IEEE International Conference on*, vol. 0, p. 78, 2002.