Master thesis in Computing Science, 20 credits

Secure Ad Hoc Networking

Claes Gahlin
<claes@cs.umu.se>

1st March 2004

Abstract

This paper focuses on the gap between proposed ad hoc routing protocols and the means to make them secure. As it turns out there currently exists no general solution but depending on the setup and use of the network there are many solutions that might be adequate.

The paper includes an overview of some of the available algorithms used for ad hoc networking, it also gives some detailed description of different cryptological algorithms that can be used to make the routing more secure. The DES algorithm is presented as is the new AES standard which provides for better security and efficient implementation. The usage of certificates is the only available solution for authentication of nodes that are practically implementable in a general ad hoc network. For the use of asymmetric cryptography the elliptic curve cryptosystem is described which proves to be more efficient than other public key systems available.

In an effort to make an ad hoc network working as the connection network to the Internet with support for mobility the use of different protocols will provide for the currently missing security. The proposed solution provides for a reasonable secure solution without much assumption on the involved nodes. By letting them share missing certificates between themselves without the need for a prior route and without flooding the entire network the proposed solution seems to be a viable solution. It provides for an efficient and quite secure routing algorithm based on AODV and transparent network roaming by using modified Mobile IP.

Contents

List of Figures

Acknowledgements

This thesis has come to be, much with the help of several people around me. First of all I would like to express my gratitude to Matilda Östling for her help with proofreading and support. I would also like to thank Mr. Fredrik Stålnacke for his many insightful comments that has helped me see some things that otherwise would have been overlooked.

For his good answers to my questions I would like to thank Mr. Christer Åhlund and finally also Dr. Jerry Eriksson who has been my tutor and inspired me to research this subject.

1

Introduction

This chapter provides a short summary of the work presented in the paper and also some pointers of what each of the following chapters include.

1.1 Ad Hoc Networks

An ad hoc network is a network that often is based on mobile devices. There is no need for existing infrastructure since each node can act as a router and forward traffic between other nodes. The name mobile ad hoc network (MANET) is often used to describe these networks. Since there is a lack of infrastructure and the node mobility is way larger than in wired network new routing protocols are proposed to handle these new challenges.

With the nature of ad hoc networks was the idea of on demand routing protocols born and these have been shown more efficient in both computational and communicational resources. Although often less efficient there exists algorithms which are more like the ones that are used in the wired networks. These provide faster routing but at the cost of more computational resource or periodic updates of routes that are seldom used.

1.2 Security and Cryptography

To secure information many different approaches have been proposed over the years. The ones that have gotten the most common use are the Data Encryption Standard (DES) and the Rivest-Shamir-Adleman (RSA) Cryptosystem. The DES system is getting older and the new standard, the Advanced Encryption Standard (AES), is getting deployed in more and more solutions.

The usage of cryptography depends on the ability to manage key distribution which has been shown to be the largest problem. With the advent of public key systems there were solutions proposed based on globally available directories to solve the key management problems. The directories never got going really, but similar solutions exist based on distributed trust and certificates.

In the wireless domain a number of additional attacks are possible just because of the used ether. For the case of ad hoc routing the number of problems are even

higher. The availability of the transmitted traffic adds vulnerabilities that are not present without a larger effort in the case of wired networks.

Most physical aspects such as radio jamming will not be considered in this paper. Solutions using frequency hopping or spread spectrum signalling has been proposed in other work to defend against these kinds of attacks. However, physical security related to mobility and ease of theft will be reviewed.

1.3 Thesis Outline

In chapter 2 the ideas behind ad hoc networks are ventilated together with overview descriptions of different routing protocol proposals. The descriptions are based on common network routing paradigms that are shortly described and compared against. Finally some remarks are made about the problem of ad hoc routing in contrast to regular networks.

A short introduction to security in general is given in chapter 3. The chapter also includes some terminology and taxonomy of different kinds of attacks and threats.

Since a lot of the security is accomplished using cryptographic algorithms chapter 4 introduces the concept of cryptography. Moreover, the currently most common algorithms used for different cryptographic tasks are described. This is a highly theoretical chapter that might demand some mathematical background. However, it should be readable to get an overview of what is currently possible in the field.

The large chapter that follows goes into detail of different security aware protocols suited for use in ad hoc networks. Chapter 5 has a large portion of protocol descriptions together with some general discussion of the usability of the protocols in different scenarios.

A concrete case study is made in chapter 6 where a system for extending global IP connectivity using ad hoc connection networks is described. The proposed solution is based on mobile IP and ad hoc networks in a close collaboration. The chapter describes the setup briefly and finishes of with a discussion about possible security enhancements.

Chapter 7 concludes the paper and discusses some further research contexts for future exploration.

2

Ad Hoc Networks

In this chapter the concept of ad hoc networking is introduced. The questions of when this could be useful are also addressed even though it will not be completely drained. There is a large potential in future applications to use it. Furthermore the challenges in ad hoc networking are up for discussion together with short description of the commonly used algorithms for routing.

2.1 What is Ad Hoc Networking?

Users of connected technology, which is an ever growing number of people and machines, are getting more and more accustomed with constantly being able to have access to different on line services. It may be e-mail or on line dictionary services, ticket booking, or travelling information such as road maps and driving directions.

The networks for mobile phones are usually available in most inhabited areas. As new technology is developed even more services are available through these networks. Even high capacity hot spots are being common practise in densely populated cities, mostly at hotels and airports even though some gas stations are getting hot-spots installed. This will of course continue to expand to more areas.

Both techniques used for wireless connection is still dependent on base stations to connect to. These base stations are in turn connected to an infrastructure. To be able to expand the wireless services they depend on this infrastructure. Problem arises when such infrastructure is not available. It can be costly to build new links for which there is little profit for the service provider.

What we would like is the ability to connect to available services without the need for an infrastructure. These spontaneous connections are, as the name implies, the ad hoc part of the networking.

Solutions exist for mobile users to connect to each other through the Internet. This can be accomplished using DHCP[1] and Mobile IP. This does, however, depend on the availability of servers that allow the users computers to connect. Using Mobile IP (MIP) the information between two users in the same room even gets routed and tunnelled through the Internet.

[1] Dynamic Host Configuration Protocol, RFC 2131, Mars 1997

Throughout the history of ad hoc networks they have also been called network on demand or mesh networks. Although given these names they are of similar operational ideas. The ad hoc networks have been on the research desk for a long time but have recently gained more interest. The military applications of such networks have seeded new interest into the research community. The name ad hoc networking is used trying to demilitarise the jargon. Initially this type of networking was researched by the military and was called packet radio.

2.2 When is Ad Hoc Needed or Used?

There exists numerous occasions where an infrastructure is not available. This section will present some of the example scenarios where ad hoc networks might come in handy. The basic ideas are possible commercial usages exemplified by Perkins [20].

2.2.1 Emergency Services

Anywhere when there is an emergency there is a need to co-ordinate the rescue personnel. This is commonly solved using hand held or vehicle mounted radios. However, what about the infrastructure that may have been damaged and is no longer in operation? To quickly get things going again the use of ad hoc networks can automatically fix this.

This might not be such a big problem in small fires or so, but when larger areas are hit by a natural disaster it can be important to quickly be able to communicate. By using ad hoc networks to set up a network infrastructure it is simply a matter of placing out a couple of mobile routers which makes it easy and fast.

2.2.2 Conferences

In many situations the need for connecting and exchanging information between participants of a conference or some other meeting is clear. Usually there is a great need for collaboration and since the home network environment is not available there is a need for other solutions.

There are usually available networks for the participants to use but this might imply very large round trips for the data using for example Mobile IP [20, 28].

2.2.3 Home Networking

Given that the use of wireless computers and appliances keeps on growing in the home environment the need for helping out administrating this is also expanding. Using the techniques of ad hoc networks that configures themselves is truly something that would be of great help.

Also, if the computers are used at more places than at home, at the office or school maybe, there is a still larger administrative burden that must be kept down.

2.2.4 Personal Area Networks

Many objects that are tightly coupled to a single person can take advantage of being connected to each other forming a personal area network. The network itself is most definitely mobile since people tend not to stay around for long in one spot. This makes the use of ad hoc in personal area networks less needed. However, when getting connected to another personal area network (PAN) the connections between persons devices might be wanted. In this case there is definitely a need for ad hoc networking support.

2.2.5 Embedded Systems

As more and more machines everywhere is in need for communicating different things to the surroundings a need for ad hoc networking arises. One can think of objects that can respond to changes in the environment and together with other devices perform different scenarios depending on the current context.

It might be a toy with built in networking capabilities that can interact with the home computer to lookup some data at the Internet or a connected phone that can turn down the volume of the stereo and TV when there is an incoming call.

Some researches are thinking about ubiquitous computing, where we will have computers connected to each other performing tasks depending on changing environment all around us. It is hard to see every possible use of this technology at the current time, but new services and applications will surely benefit of having ad hoc network support.

2.2.6 Sensors

Using tiny devices that are able to gather different information such as temperature, concentrations of different chemicals and gases, vibrations, and so on can be of importance in accidents and emergency situations. Constructing these sensors so that when turned on they form an ad hoc network and report back to a well known data collecting node they can be of great importance.

For example in the case of a gas leak, instead of sending rescue personnel into the dangerous area these sensors can be dropped from an air plane or helicopter. The use of the gathered data can be helpful in devising a plan to take care of the situation.

In the military field there can of course be a lot of applications for these kinds of data collecting devices.

2.3 Routing

The routing of data packets in computer networks is the process of moving information from a source to a given destination. The way to do this in the best way

possible is the concern of the routing algorithm used. In the case of wired networks the main routing algorithms used are either distance vector routing or link state routing.

2.3.1 General Routing Principles

Routing has been going on in large networks, as the Internet, for quite some time and lots of different routing algorithms has been proposed and tested. The main categories that have survived and have been in large use are those described below. These are performing well in wired networks, but some problems still exists.

Distance Vector Routing

A distance vector routing algorithm is a distributed algorithm that is quite simple to implement and has low computational demands. It basically means that each router has a way of knowing about its neighbours. The router knows the metric of each link to those neighbours. The metric can be of any type, for example delay or similar.

Also, each router has a distance vector, that is, a table of distances, to each destination available and which outgoing link to use. All the neighbouring routers exchange information between each other based on what they know about their neighbours and what they told them. Each time a router receives an update from a neighbouring router it updates its own vector with the minimum distances. If some values are changed the router in turn, sends out an update to all its neighbours.

Distance vector routing is commonly known to have problems with changing topology. Especially a problem known as the count-to-infinity problem which makes the routers construct an ever growing path to a node that has gone down or a link to that node being broken. The algorithm can also converge slowly on changing topology, and while converging, create routing loops [34].

Link State Routing

Instead of depending on every neighbour to gradually give information about how to get to all the destinations the link state routing algorithm gets a complete picture of the entire network and locally calculates the shortest path to every destination. This means that for each router to get information about every link all the routers must broadcast (flood) the information that they have to all other.

When all the information needed is collected the process of finding the shortest path can begin. This is accomplished using Dijkstra's or Prim's algorithm. To be able to compute the shortest path all nodes need to have a complete view of the network and then performs the computations locally. This makes these kinds of algorithm global routing algorithms in comparison to the distributed or decentralised distance vector algorithms.

As has been stated this kind of algorithm demands a complete view over the network which makes the number of messages needed to be sent between all

routers relatively high. Also, the computation of the shortest path problems using Dijkstra's algorithm is in the order of $\mathcal{O}(n^2)$ [14].

Comparison

As was described above the distance vector routing algorithms are distributed, relatively easy to implement, and does not need much processing power or memory. In the contrary, link state routing demands more bandwidth to distribute all information between all routers, more memory to hold the complete topological graph, and more processing power to compute the final result.

However, the distance vector approach does have some problems with slow convergence on changing topologies. There are of course some problems with the link state variant also, for example how to know when all information has been collected and whether all nodes are working with the same overview or are doing their calculations on different topologies.

What is gained with the link state approach is robustness. If one router gets the wrong idea of the network topology or calculates the wrong routes it can damage some paths. But if a router using the distance vector algorithm gets the wrong idea about some link status it is spread to all the other routers and they have no idea what is right or wrong [14].

Scaling and the Hierarchical Approach

There is a problem with both mentioned algorithms, they scale relatively bad. The algorithms described above need a distance to all possible destinations. On large networks this is a big problem. But there are solutions.

By dividing the network into different areas and layers it is possible to route the messages within each area separately. If a message needs to go to another area it is forwarded to an area gateway which in turn routes messages between the other area gateways and so on. Using this technique it is possible to cut down the number of destinations each sub level router needs to know about thus saving memory, bandwidth, and processing power.

2.3.2 Routing in Ad Hoc Networks

In the case of a mobile ad hoc network the topology is highly dynamic. This leads to quickly changing link states. Some links get broken while other links are created by other pairs of routers as is depicted in figure 2.1 on the following page. In this picture the mobile host 1 (MH_1) is moving from the vicinity of MH_2. As it gets closer to MH_7 and MH_8 new links are established to these hosts. These characteristics are different from the one that appears in most wired networks. The routing algorithms used in the wired case have problems with topology changes, and if these happen often the problems are just getting worse.

Another problem that arises in wireless networks that is not as common in wired routing is the asymmetrical links. That is, one node can reach another but

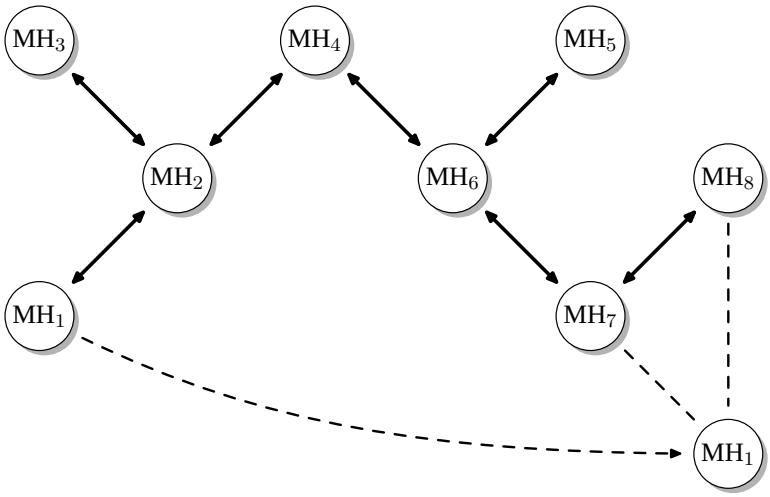

Figure 2.1. *An ad hoc network of mobile nodes [20].*

the return path is not the same. Some ad hoc routing algorithms described below handles this and some do not.

The following sections describe different solutions and main ideas about routing in ad hoc networks that overcome, or at least damps, some of the routing problems. This is not intended as a reference of routing protocols, but instead a short introduction that emphasises the different needs.

Destination-Sequenced Distance Vector Protocol

Destination-Sequenced Distance Vector (DSDV) is an ad hoc version of the commonly known distance vector algorithm. To overcome the problems with slow convergence in the ordinary distance vector algorithm and prevent routing loops in highly dynamic topologies this algorithm adds a sequence number to the routing table entries.

The sequence numbers are updated by the destination nodes when new links to them are detected. Also, when a node detects a broken link it sends this out together with an updated sequence number. The receiving nodes check for higher or equal sequence numbers. If a route update packet is received with lower sequence number it is discarded. In the case of a sequence number that is equal to the one already held the metric is checked to see if it is better or worse.

To save bandwidth the protocol uses two types of route update messages. First one is a full dump that a node sends to all its neighbours. These messages contain the complete routing table. The other variant is an incremental update which only updates the routes that has changed since the last full update. In this way it is possible to send only small packets, conserving bandwidth and transmission time for most cases. When these incremental updates are getting too big the node can

send a full dump instead in hope to be able to send smaller packets after that.

To get rid of the possibility of an oscillating system update messages are only sent out after a delay. After this delay the routing information has stabilised and is not that sensitive to oscillation. The delay is computed using a running, weighted average over the most recent updates.

Since the topology might change the route update messages are sent out at certain time intervals. However, there is no need for synchronising the different nodes as the update events are handled asynchronous. The use of these repeating update messages keeps all the nodes busy when not in need for communication. However, when the needs arrive all nodes are ready to forward the data directly.

The DSDV algorithm gets rid of the undesirable properties that the original algorithm possesses. It propagates the bad news of broken links fast and keeps the path updates stable. Also, using the sequence number rules for updating distance vector values it guarantees loop-free paths to each destinations at all times [22].

Dynamic Source Routing

The Dynamic Source Routing (DSR) protocol is completely demand based. It does not need any kind of periodic updates or node announcement messages. Instead, the protocol acquires the needed routing information on demand.

The routing protocol is divided into two parts. The first is the route discovery and the second is route maintenance. The discovery phase is initiated when a node needs to send information to another node that is not available in its current path cache. The node broadcasts a special discovery packet with the destination and a unique identification number. The packet is received by all nodes within the wireless transmission range. They, if they are not the destination, add their node address to the path in the packet header and retransmit the discovery packet. A packet with the same identity as has already been seen is discarded. Also, if the node itself is mentioned in the path header the packet is discarded. This technique efficiently cuts down on duplicate packets in the air.

When the packet finally finds its way to the destination, the destination node returns a route reply. The reply is sent using the routing cache if present. Otherwise a new route discovery is initiated but in this case with the route reply piggy backed on the discovery packet. If this piggybacking is not allowed there is a great risk of infinite looping of route discoveries. Another way is to reverse the source path collected by the route discovery; however, this takes for granted that all links are bidirectional which may not always be the case. A simple example of the broadcast of source route discovery packets is shown in figure 2.2 on the next page.

After a path has been discovered the second phase of the protocol is in use. This is the maintenance phase. During this phase all communications are done using the previously found paths. Each node on the path is responsible for resending packets that are not acknowledged by the next hop node. After a maximum limit number of retries a route error message is sent back to the source node indicating that the path is broken.

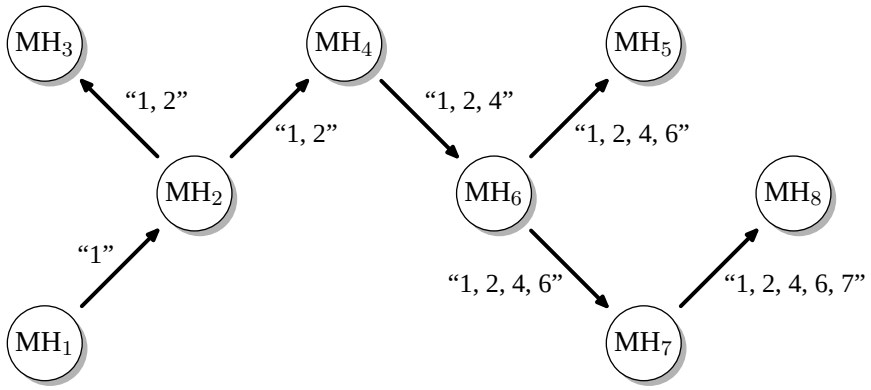

Figure 2.2. *DSR route discovery example with mobile host 1 as the initiator and 8 as the target.*

A number of different optimisations of the protocol are possible. For example, each node on a path that is not actually originating a route discovery can cache some part of the messages that they see go by. This can speed up route discovery but may also place some overhead on the software and the processing time of the processor [11].

Ad Hoc On-Demand Distance Vector Routing

In contrast to the DSDV, the Ad hoc On demand Distance Vector (AODV) is an on demand protocol. It borrows the idea of route discovery and maintenance while still using a distance vector approach to the routing.

Each node keeps a distance vector for each known destination and its next hop neighbour. If a destination that is needed is not in the list a route discovery is initiated. By issuing a broadcast packet to its neighbours containing source and destination addresses and a sequence number the node hopes to find a path to the destination.

If a node, that is not the actual destination, receives a request it checks its own routing table. If it can find the destination there it replies to the source node with this information. If, however, this information is not in the node's routing table it adds the source as a destination in its own table using appropriate hop count, increments the request packet's hop count and rebroadcasts it to its neighbours.

This procedure is continued until the destination is reached and a route reply packet is sent back to the source. After this all data traffic is routed using the discovered path. If the nodes move and some links break a route error packet is sent out to tell the nodes that the path is no longer available. If this happens the source node initiates a new route request. An example of a route discovery and path setup is shown in figure 2.3

To eliminate the need of flooding a large network with route request packages the time to live field is used to limit the number of hops a route request can be sent. If a request fails this is gradually incremented until the destination is reached. This

	Route Request			**Route Reply**	
1	$S \rightarrow *:$	$\langle \text{RREQ}, S, D, 0, S \rangle$		$D \rightarrow C:$	$\langle \underline{\text{RREP}}, S, D, \underline{0}, \underline{D} \rangle$
2	$A:$	$d[S] = 1, n[S] = S$	6	$C:$	$d[D] = 1, n[D] = D$
	$A \rightarrow *:$	$\langle \text{RREQ}, S, D, \underline{1}, \underline{A} \rangle$		$C \rightarrow B:$	$\langle \text{RREP}, S, D, \underline{1}, \underline{C} \rangle$
3	$B:$	$d[S] = 2, n[S] = A$	7	$B:$	$d[D] = 2, n[D] = C$
	$B \rightarrow *:$	$\langle \text{RREQ}, S, D, \underline{2}, \underline{B} \rangle$		$B \rightarrow A:$	$\langle \text{RREP}, S, D, \underline{2}, \underline{B} \rangle$
4	$C:$	$d[S] = 3, n[S] = B$	8	$A:$	$d[D] = 3, n[D] = B$
	$C \rightarrow *:$	$\langle \text{RREQ}, S, D, \underline{3}, \underline{C} \rangle$		$A \rightarrow S:$	$\langle \text{RREP}, S, D, \underline{3}, \underline{A} \rangle$
5	$D:$	$d[S] = 4, n[S] = C$	9	$S:$	$d[D] = 4, n[D] = A$

Figure 2.3. *Simplified route discovery example in AODV. Host S is the initiator of the route request (*RREQ*) and the destination is host D.*

approach is called expanding ring search.

The AODV protocol can only handle symmetrical links but has the ability to also be used in multi cast groups. Additionally it supports a hello message, a variant of the route request message, which is used to detect the one hop neighbours. This may not be used if the lower layers already support this kind of service [23, 21].

Cluster-Based Networks

In some situations the network may be small, that is, consist of a small number of nodes. In this case the above mentioned routing algorithms may be adequate. However, if the network grows and the number of nodes are large there may be unnecessary overhead and the network diameter, the number of hops, may become too large to handle straight on.

In the case of wired networks area controllers are used to manage a limited set of computers and routing between these are done using some backbone infrastructure. This makes the network manageable for each smaller cluster and the network diameter might become smaller using the backbone routers. However, the setup of these area controllers is done by administrative personnel often during the design of the network. This is clearly not something that can be done as easily in mobile ad hoc networks. There are some ideas and techniques available that manages to accomplish these goals automatically using distributed algorithms and elections of cluster heads.

The idea of using clusters is beneficial in many ways. It can be used to group nodes into clusters that are separated to nearby clusters by transmission frequency or spread spectrum coding. This way the congestion can be lowered [32].

Zone Routing Protocol

A common approach in many fields is to take the best ideas and combine them. This is one such idea. The zone routing protocol (ZRP) uses a proactive routing variant within a limited zone defined by a small number of hops around each node. Outside this zone a reactive protocol is used.

Proactive means that all the routes are known by all the nodes before any data is needed to be sent. The link state and distance vector protocols are based on these assumptions. This is useful in the ZRP because of the limited number of nodes in each zone. These zones are locally defined by every node and thus highly overlapping. Also, the idea is that there is more often communications taking place within a relatively small geographical area than to distant nodes. This may be true in some setups, for example military tactical units or emergency rescue teams, while other applications might have different demands.

The mobility of nodes affects are only local since the routing between nodes are done reactively or on demand, like AODV or DSR. The routing to a distant zone, therefore, is not affected by local link movements as much as for other protocols.

The implementation of the two different routing protocols needed can be any pair of protocols. The two variants are called Interzone Routing Protocol for the routing between zones and Intrazone Routing Protocol for the routing within a local zone [7].

2.4 Challenges With Ad Hoc Networks

As we have seen in the previous sections there are many different approaches to the routing dilemma in fast changing topology networks as mobile ad hoc networks. Even though there are a lot of different approaches to consider they fit well into different needs.

One thing that has not yet been surfaced is the need for addressing. The routing protocols are working with unique node addresses, for example IP numbers. These addresses must, however, be handed out in some way. Also, the need for gateways to wired networks needs to be considered in the addressing schema.

A big challenge is of course to keep the routing tables needed up to date with a fast changing topology. Also the problem of loop freedom and scarce bandwidth available puts even higher demands on the routing algorithm. On top of this the size of the networks can be from just a few nodes to over hundreds of them making the routing algorithms sensible for some scaling problems.

On the commercial market the use of ad hoc networking techniques have only started to be available in some networks. Most notably within the companies own wireless networks or building to building links. The use in more everyday products such as mobile phones have low commercial interest since the operators of the networks probably will lose some of the traffic. Also, not every one is happy with having their mobile phone forwarding traffic for someone else. Not because of the traffic itself but because the battery power drain. The largest possibility for

the commercial breakthrough is probably within the wireless local area networks where the standards already has support for some limited ad hoc connections, or peer to peer support.

The use of multi hop wireless networks can help keeping the power consumption down due to lowering the link length. However, the need to make the routing protocols power aware and not waste too much power on control messages instead of actual information traffic is essential. Also, multi hop networks are dependent on the intermediate nodes being available even though that node may not be in transceiver mode. The use of multiple available routes might be a solution where some nodes can go down in power saving mode while others peek up now and then to sense the communications.

When all the traffic start to flow around between everybody seemingly uncontrolled the need for security and authentication arises. This is what the rest of this thesis will focus on.

3

Security Model

The field of security is large and some model to use for attacking the problem is needed. In this chapter some attributes of a secure system will be presented together with an introduction to the types of attack that are available to the malevolent user.

3.1 Security Attributes

To be able to classify the different security needs of the applications of ad hoc network the following attributes needs to be considered [37].

Confidentiality Confidentiality is the process of keeping the information sent unreadable to unauthorised readers. Since the wireless medium is more available not only to its intended audience but also to eavesdroppers the need for securing information traffic from end to end is becoming even more important.

One way to reach confidentiality is the use of cryptographic techniques which will be discussed in later chapters.

Authentication Not only does it matter to keep the information safe from eavesdroppers or otherwise unauthorised readers. The need for knowing that the sender actually is the sender is important for secure systems.

In some cases this need is even more important than the one of confidentiality. It might be of more importance to know that a certain message actually is from the one who it says it is than to keep the information it holds secret.

Availability Availability is important in many different applications. Some attacks against the availability are the well known denial of service attacks where many thousands of phony requests to web servers and the like drown the actual service.

A service that is secure must also be available to the one ones who depends on it. Otherwise there is no need for having the service at all.

Also, in the wireless arena when hand held and portable devices are in great use the risk of thievery is large and also fits into the category of availability.

If the theft has already taken place, then the authentication property and end to end confidentiality might be threatened even more.

Integrity When the information gets sent along channels the risk is that attackers see a message, change some important data and resends it. The integrity is the ability of the secure system to guarantee that the received message is in fact the real one that has not been tampered with. Also, some form of replay attacks might threaten the integrity attribute.

Non-Repudiation Along the lines of message sending and receiving the non-repudiation attribute is the ability not to be able to deny sending a message. This may be of great importance in some situations but might not be in some others.

In addition the availability attribute adds a number of criteria on the routing protocol in use. To achieve availability and security in a routing protocol the following criteria should be met according to Lundberg [15].

Certainty of discovery This criterion means that if a route between two nodes exists it should be found. In addition, the found route must of course be to the correct node.

Isolation It should be possible to identify and isolate misbehaving nodes. Alternatively, the routing protocol should be resistant to malicious nodes.

Lightweight computations The need for lengthy and demanding computations should be avoided if possible. If not, the computations should be affecting as few nodes as possible. The preferable situation is of course that only the peer nodes needs to be involved in the computations.

Location privacy The location of nodes should not be available for a malicious user as this information might help in other kinds of attacks or be just as important as the information in the data messages. An attacker should not be able to get hold of information regarding node location.

Self-stabilisation If an attacker has been able to inject forged information into the routing paths the routing protocol should be able to recover during a finite amount of time. The property also says that if an attacker is to permanently disable the network he must remain around and continue to send forged information. This way, it might be possible to locate the malicious nodes.

Byzantine robustness The Byzantine robustness property is even stricter than the self-stabilisation property in that the routing protocol should not only be able to recover from an attack it should be able to function properly during an attack. For a description of Byzantine failures see [18] and the Byzantine generals' problem.

3.2 Threats and Attacks

The number of different threats and attacks can be categorised into a number of different areas that they target. The first is to consider the level of the attack which can be perceptual where the human perception is targeted using the media as a bearer. It may be broadcasting false information or just observation of social behaviour to be able to alter decision processes.

Secondly the attacks can target the information itself where interception and eavesdropping comes naturally in thought. Of the more active nature of these attacks might be the creation of false messages injected into networks. Also the denial or degradation of network services is a form of active attack on the information level. In this category application level attacks such as Trojan horses or viruses and the like are also included.

The physical attacks are the third category. The passive nature of this category can be radiation interception or inductive wiretapping. The more hands on attacks include theft of equipment, cryptographic or physical keys, and different storage medias. Other kinds of attacks are social engineering or as drastic as destruction using explosives or other physical force [18].

3.2.1 Wireless Network Attacks

In contrast to network equipment in wired networks where the devices usually are kept behind locked doors the ad hoc network equipment are usually carried around as small battery-powered devices or placed inside mobile units like cars. This makes them even more attractive for attackers since they are often easier to get to and also easier to carry away from the crime scene. Another point is that it can be quite hard to intercept wired media without getting noticed both because the media itself might be hard to get to and to intercept the cables often will need cutting the cables for a while. In the wireless medium it is as easy as just putting up an antenna, usually small enough not to be noticed.

Also, since many users of the ad hoc networks will be using it in public places the threat of unintentionally revealing secrets are large. This can be in the form of a conversation being held so that someone can overhear secret information or shoulder surfing, that is, someone reading the computer screen or keyboard from behind while entering passwords or the like. The human nature of bad memory can also be of some help for the attacker. It is not uncommon that individuals write down passwords and user details on post-it notes and at a later time throw them away in garbage cans. The retrieval of this kind of information can help attackers to guess the correct passwords to system resources. This kind of attack has gotten the common name of “dumpster diving” [18].

3.2.2 Attacks on Ad Hoc Networks

In addition to often being wireless the structure of an ad hoc network, or lack there of, leads to some special kinds of attacks. Especially attacks on the connectedness of the network which means attacks on the routing protocol. In this section some of these attacks will be addressed.

Routing Loop

By sending forged routing packets an attacker can create a routing loop. This will result in data packets being sent around consuming both bandwidth and power for a number of nodes. The packets will not reach their intended recipient and thus can be considered a sort of denial-of-service attack.

Black Hole

The setup for the black hole attack is similar to the routing loop attack in which the attacker sends out forged routing packets. It can setup a route to some destination via itself and when the actual data packets get there they are simply dropped, forming a black hole where data enters but never leaves.

Another possibility is for the attacker to forge routes pointing into an area where the destination node is not located. Everything will be routed into this area but nothing will leave also creating a sort of black hole.

Grey Hole

A special case of the black hole attack is an grey hole attack. In this attack the adversary selectively drops some kinds of packets but not other. For example the attacker might forward routing packets but not data packets.

Partitioning

Another kind of attack is for the attacker to create a network partition in which some nodes are split up to not being able to communicate with another set of nodes. By analysing the network topology the attacker can choose to make the partitioning between the set of nodes that makes the most harm into the system.

This attack can be accomplished in many kinds of ways. Both by forging routing packets as in the previous attacks but also using some physical attack such as radio jamming.

Blackmail

Some ad hoc routing protocols tries to handle the security problems by keeping lists of possibly malicious nodes. Each node has a blacklist of, what it thinks, bad nodes and thereby avoiding using them when setting up routing paths. An attacker

might try to blackmail a good node causing other good nodes to add this node to their blacklists and so avoid it.

Wormhole

In the wormhole attack an attacker uses a pair of nodes connected in some way. It can be a special private connection or the packets are tunnelled over the ad hoc network. Every packet that one of the nodes sees are forwarded to the other node which in turn broadcast them out. This might create short circuits for the actual routing in the ad hoc network and thereby create some routing problems.

Also, all the data can be selectively forwarded or not using this attack thereby controlling the ad hoc network to a large extent. This kind of attack together with a partitioning attack can gain almost complete control over the network traffic.

Rushing Attack

Many reactive routing protocols keep a sequence number for duplication suppression at every node. An attacker can distribute a large number of route requests with increasing sequence numbers forged to appear to be from other nodes. This way when the actual route request is sent out many nodes suppress it as a duplicate and thereby disrupt the actual route discovery.

Resource Consumption

By injecting extra data packets into the ad hoc network limited resources such as bandwidth and maybe battery power are consumed for no reason. Even more resources might be consumed by injecting extra control packets since these might lead to additional computation. Also, the other nodes might forward control information as it comes in resulting in even more resource consumption [9].

For devices that tries to conserve battery power by only occasionally enable their communication device a malicious attacker might communicate in an ordinary way but with the only intent to drain battery power. Stajano and Anderson call this resource consumption attack “sleep deprivation torture” [30].

3.2.3 Cryptographic Attacks

The common use of cryptography has also been targeted for different kinds of attacks. These are of course often dependent of the previous attacks of eavesdropping or thievery. The main target for the attack is to find the plaintext or the secret key, or both. Due to Kerchoffs it is generally assumed that the attacker is aware of the algorithm in use why all the security should be in the knowledge of the key.

Passive Attacks

The passive attack is mainly aimed at recovering the plaintext or the encryption key. The cryptographic attacks can be categorised according to the level of information that is needed, or available, for the attack to be successful. These general attacks are according to Schneier [29]:

Ciphertext Only Attack: In the case of the known ciphertext attack a cryptanalyst only has information regarding the ciphertext and the algorithm. The job is to find the plaintext from this information, or as much of it as possible. If no weaknesses are known about the actual algorithm a brute force method must be applied in which all possible keys are tried to finally get to the actual message.

Known-plaintext Attack: In this attack the cryptanalyst has not only knowledge of the ciphertext but also to the corresponding plaintext. In this case the job is to find the key that has been used to encrypt the messages. This attack might not be as unimaginable as one might think at first. Many messages contain common structure that is easy to guess. For example most letters have the date sent and the sender identity included in the plaintext, information that the attacker may find out easily.

Chosen-plaintext Attack: In a chosen-plaintext attack the cryptanalyst can choose the actual plaintext messages to have certain structure that will help him or her to gain additional information about the structure of the key. This is more powerful than the known-plaintext attack.

Adaptive-chosen-plaintext Attack: This is an even more powerful attack based on the chosen-plaintext attack. In this attack the cryptanalyst not only chooses the whole message, he or she can choose smaller parts based on the previous results.

Physical Attacks

Some algorithms implemented in hardware as in smart cards have been shown to be vulnerable to a more physical kind of attack. Using timers it is possible to measure the time differences on the encryption of messages using different keys. This way it is possible by analysing the time to deduce the actual key that was used. These attacks are commonly called timing-attacks.

Another variant on the same theme is that of power-attacks. Certain processing units draw a different amount of power depending on the actual operations being performed. By measuring the power usage of the processing unit it will be possible to analyse the actual operations and operands and thereby infer the key or plaintext.

Active Attacks

Active attacks are mainly based on the different passive attacks but include some more active measure of the attacker. These kinds of attacks are harder to defend against. The chosen-plaintext attack might for example be accomplished by sending a chosen message through an encrypted channel that the attacker is able to eavesdrop.

Other active attacks are *rubber-hose attacks*, as Schneier [29] calls them, including threatening, blackmailing, or torturing some party in order for them to give you the key. Bribery is another such attack.

3.3 Taxonomy of Attackers

Hu, Perrig, and Johnson [9] propose a set of attacker models in ad hoc networks based on the nature of the attacking capabilities. They devise the taxonomy of passive and active attackers where a passive attacker only eavesdrop while the active attacker takes an active role in the attack such as injecting or forging packets.

Since the passive attacker has a low safety violation the active attackers are further classified. The classification is based on the capabilities and number of adversaries. An active-n-m attacker is an attacker that has compromised n nodes and owns m. The authors also propose a hierarchy with increasing strength of these attackers as:

- Active-0-1 where the attacker owns one node.
- Active-0-x where the attacker owns x nodes, $x > 1$.
- Active-1-x where the attacker owns one compromised node and distributes the cryptographic keys to its $x - 1$ other nodes.
- Active-y-x.
- Active-VC where the attacker has compromised all the nodes on a vertex cut through the network thereby partitioning the network and controlling all the traffic between the partitions.

4

Cryptography

"Cryptography is the art of writing in secret characters." *Webster's Unabridged Dictionary*. That is, to hide information from an unauthorised party. The process of making messages unreadable dates a long way back in time. From its initial and limited use by the Egyptians some 4000 years ago [16] until today where it is part of everyday web browsing.

The modern approach has a solid base in mathematics and especially number theory. This chapter is about the art of concealing information. In the introductory part some classical ciphers are brought up to give a good insight in the development over time. Later the most commonly used ciphers are presented. These are the cornerstones of secure transmission over insecure channels in use today.

4.1 Introduction

To get a deeper understanding of the subject some of the first, and very simple, cryptographic ideas are introduced here. These are not in any common use these days but will function as good examples of how the evolution of cryptology has come to its position today.

4.1.1 Who are Alice and Bob?

To make it easier to discuss the workings of cryptographic algorithms and ciphers a couple of parties are involved. Instead of calling them A and B as in many other formal texts many papers and books in the area of cryptography uses common names. The major parties in communication systems are Alice and Bob [1]. A more humorous introduction and description of Alice's and Bob's whereabouts can be found in [6].

Alice wants to send a message to Bob without an opponent, Oscar, being able to understand the actual message. The message has to travel in an insecure channel. The channel can be any kind of medium such as a telephone line or computer network. The original message that Alice wants to send is called the plaintext. This plaintext can be any kind of message. Before sending the message Alice encrypts it with a secret key that only she and Bob has knowledge of. The message travels over the channel and when Oscar gets hold of it eavesdropping the channel

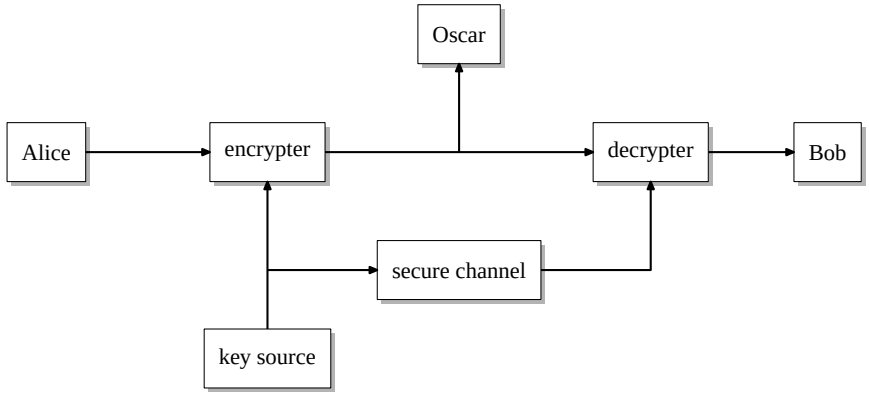

Figure 4.1. *The Communication Channel*

he can not make any sense out of it. On the other end Bob receives the message, uses the secret key and decrypts the ciphertext into plaintext. After this Bob is able to read it. The scenario is illustrated in figure 4.1 where it is obvious that a secure channel must be provided for Alice and Bob to share the secret key.

4.1.2 Simple Cryptosystems

There are two commonly known ciphers systems that most people have some clue about from their childhood. These are the shift cipher and the substitution cipher.

To begin with the shift cipher it is based on modular arithmetic. That is, given a plaintext character shift the output a number of steps, this is the key, with wraparound on the alphabet in use to obtain the ciphertext. With the secret key $K = 3 \in \mathcal{K} = \mathbb{Z}_{26}$ the plaintext "abc" would be encrypted as "DEF". In this example $\mathbb{Z}_{26}$ is used as an encoding for the English alphabet where each number corresponds to a character in the alphabet starting at A. Also note that the plaintext is written with lower case characters and the ciphertext uses upper case just to make the examples easier to read.

The encryption function for this simple example would be $e_K(x) = x + K \pmod{26}$. For further references on modular arithmetic consult any introductory literature on discrete mathematics or number theory, see also [12, 33]. The decryption function is as easy only using subtraction instead: $d_K(y) = y - K \pmod{26}$.

To cryptanalyse this cipher is an easy task, just do it brute force. This method would only have to try half of the keys in average. In the example the number of possible keys is 25. Even though this is a simple example with a limited alphabet the same is usually true for this kind of cipher.

To continue the classical cryptography exploration the substitution cipher consists of a permutation, π, of alphabetic characters. The specific permutation is the actual key that needs to be known by all parties that are to communicate together.

With the same alphabet as above we now have 26! possible keys which is a very large number. To attack this kind of cipher there is no idea to use a brute force attack on the keys, even for a computer.

The way to approach this cryptanalysis problem is the path of frequency analysis. By knowing the frequency of appearance for each character, bigram[1], trigram and so on it is possible to spot which character in the ciphertext corresponds to which plaintext character [33].

The previously shown examples are of course not valid for any real security but they do show the basic ideas about what is needed to be able to achieve this. As shown the number of possible keys must be large enough so that there is no possibility that a brute force attack might be conducted. Also, there should be no resemblance with respect to frequency or any other structure between the plaintext and the ciphertext. With these observations the upcoming sections discuss more modern and hopefully more secure cryptological systems.

4.2 Symmetrical Cryptography

Below a number of different algorithms that provide symmetric cryptography are presented. Symmetric cryptography is the use of cryptographic solutions which uses a shared secret that both the sender and the receiver need to know in advanced. This is the case that is shown in figure 4.1. In a more mathematical setting this can be described as

$$\begin{aligned} E_K(x) &\rightarrow \quad y \\ D_K(y) &\rightarrow \quad x, \end{aligned}$$

where E is the encryption function, D the decryption function, K the shared key, x the plaintext, and y the ciphertext.

4.2.1 One-Time Pad

The idea of the one-time pad was conceived by G. Vernam 1917. The technique was initially developed for telegraphic communication. The ciphertext is the bit-by-bit modulo 2 sum of the plaintext and the one-time key sequence of the same length. The deciphering is the exact same operation since sum and subtraction in modulo 2 are the same.

It was proved by Claude Shannon that even with infinite computing resources it would never be possible to separate the true plaintext from all other meaningful plaintexts using a ciphertext only attack. This gives the method its perfect security attribute that is so appealing. On the other hand, the obvious disadvantage is the need to distribute a perfectly random key sequence to everybody involved in the communication.

From these ideas the synchronous stream ciphers were gestated. Using a deterministic pseudo random number generator controlled by a secret key it would be

[1] A pair of characters, such as, TH or HE.

possible to send a plaintext messages without the need to distribute secret keys of the same length. The drawback is of course the random number generator which should be able to create a non-repeating sequence for the key stream that is to be combined with the plaintext to achieve the ciphertext [18, 29, 33].

4.2.2 Data Encryption Standard

The data encryption standard (DES) was introduced 1977 based on the IBM developed Lucifer algorithm. The final DES algorithm was somewhat altered by the American national security agency (NSA). DES is a block cipher which means it works on blocks of data instead of bit-by-bit. The block size in DES is 64 bits or 8 bytes. The encryption and decryption algorithm is identical except for the reverse ordering of the round keys. The effective key length in DES is 56 bits, although a key is given in 64 bits. One bit in every byte is used for parity checking purposes and is therefore not used in the actual encryption algorithm. This leads to a total number of $72,057,594,037,927,936$ keys. Additionally, a few of these keys are considered weak and should be avoided [18].

The security of the DES algorithm is solely based on the secret key. The algorithm is well known and has been under the eyes of many cryptologists. The overview of the complete algorithm is depicted in figure 4.2. For each block of plaintext the algorithm performs an initial permutation, which has no cryptographic value. This is shown as the rounded rectangle labelled IP near the top of the figure. The block is then split into two halves each being targeted for 16 rounds of key controlled substitutions and transpositions. Finally the two halves are rejoined and permuted inversely of the initial one ($\mathtt{IP}^{-1}$). This way of passing half the block through an encrypting function and then combine it with the other half is commonly called a Feistel algorithm after its inventor.

In each round the right hand side of the block is fed into a functional block (f) together with that round's sub key. The result of the function is combined with the left hand side of the block. This combination is the new right hand side of the data block. The new left hand side is just the previous right hand side. This can be described mathematically as follows for $1 \leq i \leq 16$:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K), \end{aligned}$$

where $\oplus$ is the exclusive or operation [33]. The f function is shown in figure 4.3 on page 28. As input it takes the data block (shown as A in the figure) and the sub key (K). Firstly the data block is expanded from 32 bits to 48 bits according to a specified pattern, some bits are duplicated. This is done in the expansion operation (E). The two 48-bit strings are then combined using exclusive or. The combined value is split in eight six-bit words each fed into a separate S-box. The S-boxes are non-linear transpositions from the six-bit value to a four-bit value. Each S-box output is concatenated forming a 32-bit value. Before the function ends the value is further permuted (P) into the final output value.

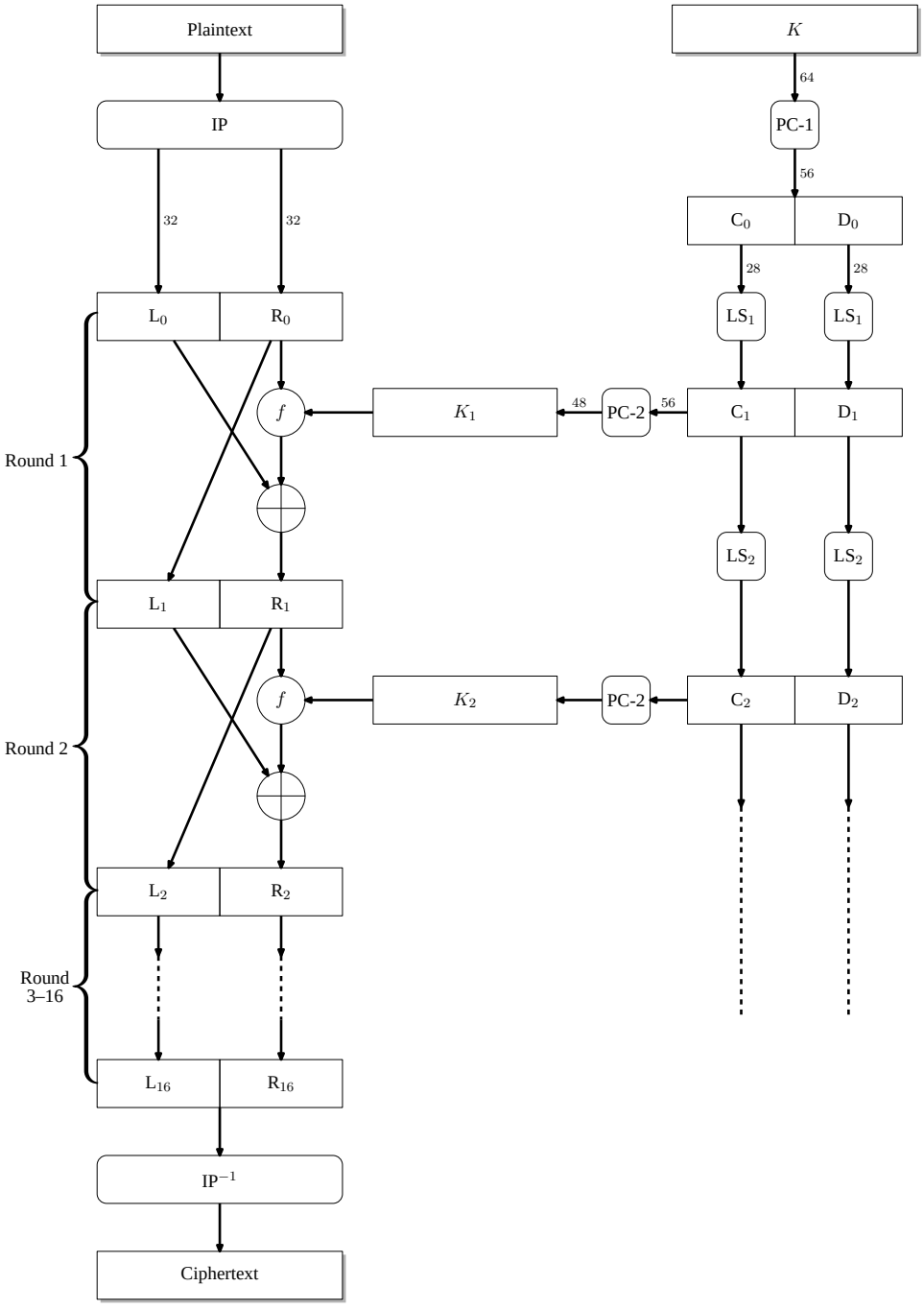

Figure 4.2. *Overview of the DES algorithm and data flow*

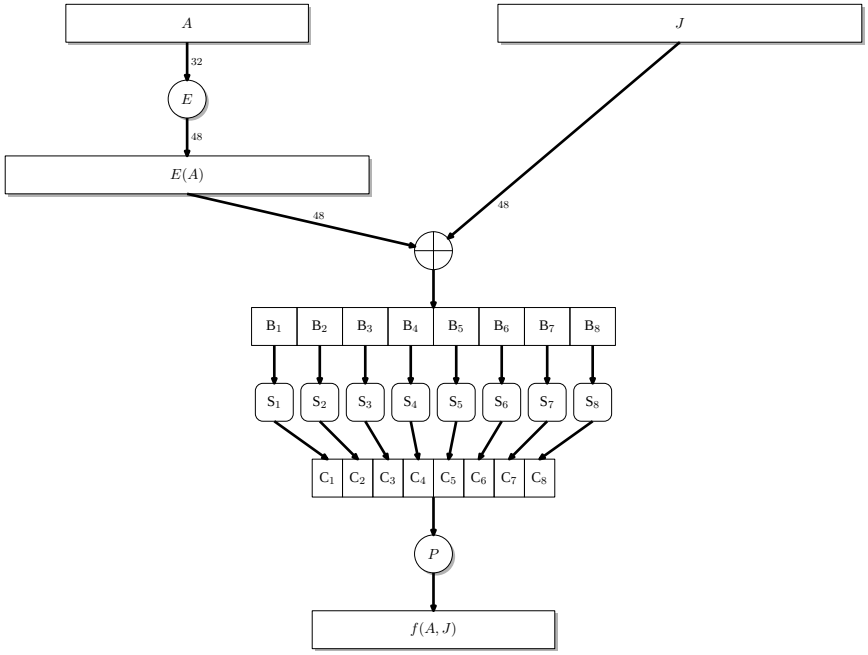

Figure 4.3. *The DES $f(A, J)$-function where A is the incoming right half (see figure 4.2) and J is the current sub key.*

For each round of the algorithm a different sub key is used. The sub key selection is shown in the DES overview in figure 4.2 on the preceding page. At the initial stage the parity checking bits are removed and the remaining key bits are permuted according to a fixed permutation. The permuted key is split in two 28-bit parts (C_0, D_0). They are then individually, cyclically left shifted by one bit. 48 bits of the key's 56 are chosen for each round. This is accomplished using a fixed permutation that only uses 48 bits of the current key value (PC-2).

Depending on the current round the cyclic left shift is done by one or two bits. It is shifted by one bit for round 1, 2, 9, and 16, for all other rounds the shift is done by two bits. Since all the permutations and number of bits to shift is fixed the key bits can be precomputed and will not need to be updated for each data block using the same key [33].

Modes of Operation

The DES describes four different modes of operation. The one described above uses the same key for each data block. This means that the encryption of some plaintext block always results in the same ciphertext. This mode is called *electronic codebook mode* (ECB) and is the common mode of operation.

A second mode is the *cipher block chaining mode* (CBC) in which the output cipher block from the previous encryption is combined with the next plaintext block using exclusive or. This mode starts with an 64-bit initialisation vector, IV. The first cipher block is defined as $y_0 = IV$. Successive cipher blocks are con-

structed using

$$y_i = e_K(y_{i-1} \oplus x_i),$$

for $i \geq 1$.

The remaining two modes of operations differ in that the encryption function is used to generate a key sequence z_i. The *output feedback mode* (OFB) is actually a synchronous stream cipher (see 4.2.4 on page 34). The key stream in OFB is created by repeatedly encrypt an 64-bit IV. Using $z_0 = IV$ the upcoming keys are generated as $z_i = e_K(z_{i-1})$ and this key stream is used to encrypt the plaintext stream

$$y_i = x_i \oplus z_i,$$

given that $i \geq 1$.

The last defined mode of operation is the *cipher block feedback mode* (CFB). In CFB we start with $y_0 = IV$ and produce the key stream using the previous cipher block, $z_i = e_K(y_{i-1}), i \geq 1$. The encryption of the plaintext is then the same as in the OFB mode. In both CFB and OFB modes the DES encryption function e_K is used in both encryption and decryption since the resulting key stream is the same.

The properties of the different modes of operation have both advantages and disadvantages. Using the ECB or OFB modes causes a change in one plaintext block to affect only the corresponding ciphertext block. On the other side, in CBC and CFB this change will not only affect the corresponding ciphertext block but also all the blocks following that one. These modes can therefore be used for authentication purposes by letting the last ciphertext block represent a message authentication code (MAC). Sending this along with the plaintext the receiver can detect changes in the message or authenticate that the message is unaltered. However, using the algorithm this way will not result in any privacy [33].

Key Length Considerations

The number of bits in the key is only 56 which, by today's measures, are a little too easy to break using brute force. Therefore, a system called 3DES is considered a replacement, at least until the new advanced encryption standard (AES) is fully deployed. Basically it allows for the use of a key length of 112 since it uses different parts of the key for each time. The algorithm is encrypting the plaintext with the first part of the key, then decrypting using the second and lastly encrypts again using the first part of the key again. This way the information has basically been encrypted three times using the standard DES algorithm but with different keys each time. This way the key length issue is no longer a problem for some time to come.

Implementation Efficiency

The DES algorithm consists of mainly basic operations that are quick to perform in both hardware and software. The only problem is the initial and final permutations that are quite tricky to get efficient in software. Some implementations just

skip this part since it does not make any difference security wise. However, this variant will not follow the standard and can not be used together with other standard conforming implementations. For hardware implementations this is as easy as hardwire the input data lines in the correct order into the next stage and will not become any overhead at all.

Since the algorithm is iterative it will need to perform 16 rounds of the operations. Each round will be some table lookups and modulo operations which are easy to implement efficient in both hardware and software.

4.2.3 Advanced Encryption Standard

The AES is based on the Rijndael Block Cipher with some restrictions on the block length which the original cipher did not have. In AES the block length is fixed at 128 bits while Rijndael can handle any block length that is a multiple of 32 bits within 128 to 256 bits. The length of the key has the same restrictions for Rijndael while only 128, 192, or 256 bits are allowed for the AES [2].

With the above restrictions in mind regarding AES the discussion that follows will consider the workings of the Rijndael cipher. During the discussion the following terms will be used:

State The State is the intermediate result that the cipher is working on. The State is initially the block of plaintext that is input to the cipher. After the encryption algorithm has done its work the State is the final ciphertext.

Cipher Key The Cipher Key is the secret key that is shared between the sender and the receiver. The different keys that are used during the encryption and decryption process are calculated from this key.

Expanded Key The Expanded Key is the key that is produced using the given key schedule and the Cipher Key. The key expansion is the process of creating the Expanded Key from the Cipher Key.

Round Key Each round of the algorithm uses a different key, the Round Key. This key is selected from the Expanded Key.

Nb The block size the algorithm is working on in words of four bytes.

Nk The Cipher Key size in words of four bytes.

Nr The number of rounds the algorithm will use as defined in figure 4.4 on the facing page.

The Rijndael cipher is, unlike DES and others, not based on the Feistel algorithm but instead built up of different layers that work on the whole intermediate state during the encryption. The cipher has a round function that is applied a number of times depending on the block and cipher key sizes. Any block size and key size may be used together. The number of rounds is given in figure 4.4. As noted

Nr	Nb = 4	Nb = 6	Nb = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

Figure 4.4. *The number of rounds in Rijndael depending on block and key sizes*

earlier in AES the only valid block size is Nb = 4 so only the first column in the table is of any importance in AES. Also, it is possible to use Rijndael using the block sizes between those given in the table.

The number of rounds is based on the fact that the diffusion is very large only after a few rounds. Moreover there are a few known attacks against a limited Rijndael using fewer rounds. The measure that has been used to decide the number of rounds is the computational effort needed to break the cipher. As long as any such attack needs at least the same amount of time that a brute force attack would have needed the attack is not considered plausible. However, as the key length increases so does the time for a brute force attack. This also allows for albeit slow attacks but more efficient than the brute force one. By adding more rounds these attacks will not likely be available.

Rijndael Encryption

First of all, before the first round the first part of the ExpandedKey, the round 0 key, is added to the plaintext block. This is done since without any key dependent operation the first round adds no security to the cipher. This is, for example, the case in DES with its initial and final permutation. After the first key addition to the State the rounds are applied.

Each round consists of four main parts and is shown in pseudo code in algorithm 4.7 on page 33. The first is the `SubBytes` function. This is a non-linear byte level substitution much like the S-boxes in DES. However this is done on each byte of the State independently. It consists of finding the multiplicative inverse under GF(2^8) and then make an affine transformation on the result, also within GF(2^8). A Galois field (GF) is a mathematical structure where a field is a ring where all elements $e \neq 0$ are units. For example, the common number systems $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ are fields [12].

The non-linear substitution is followed by a cyclic shift of the rows in the State block, `ShiftRows`. This makes each byte in each column switch to another column. This way the bytes of the State get switched around and thereby help the diffusion process. The number of steps each row is shifted is dependent on the block size. The first row is not shifted at all. The second row is rotated `C1` number of steps to the left. `C2` and `C3` denotes the number of steps to rotate rows three and four respectively. The values for these variables are given in figure 4.5.

After the rows have been rotated around the columns are mixed by acting on

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	2	4

Figure 4.5. *The Rijndael* `ShiftRows` *variables*

each column separately. Each column is one 32-bit word that is considered as a polynomial over GF(2^8). These polynomials are multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x)$ given by

$$c(x) = 03x^3 + 01x^2 + 01x + 02.$$

The polynomial is co-prime to x^4+1 and therefore invertible. The operation of this multiplication can be written as a matrix multiplication. Let $b(x) = c(x) \oplus a(x)$,

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

As can be seen in algorithm 4.7 on the facing page this operation is called `MixColumns`.

Finally each round is ended by a bytewise exclusive or operation between the State and the round key called `AddRoundKey`. For the final round the `MixColumns` operation is omitted as can be seen in algorithm 4.8.

Key Schedule

The round keys are computed by expanding the Cipher Key. The expansion of the cipher key is called the key schedule. The expansion is done differently depending on the key size. For key sizes below 192 bits the expansion is as follows. Let $W[\,]$ be a vector that holds the expanded key. Each index into the linear array is a four-byte word. The first `Nk` words are taken from the initial Cipher Key and the remaining is recursively calculated from these. For $i \geq \mathrm{Nk}$ and $\mathrm{Nk} \leq 6$

$$W_i = \begin{cases} \mathrm{SubBytes}(\mathrm{RotByte}(W_{i-1})) \oplus R_{i/\mathrm{Nk}} \oplus W_{i-\mathrm{Nk}} & , i \bmod \mathrm{Nk} = 0 \\ W_i \oplus W_{i-\mathrm{Nk}} & , i \bmod \mathrm{Nk} \neq 0 \end{cases},$$

where R is the round constant

$$R_i = (\mathrm{RC}[i], 00, 00, 00)$$

independent of Nk and RC$[i]$ is representing an element in GF(2^8) with a value of $x^{(i-1)}$. The `RotByte()` function is simple a function that takes four bytes (a, b, c, d) and returns (b, c, d, a).

```
Rijndael(State, CipherKey)
{
    KeyExpansion(CipherKey, ExpandedKey);
    AddRoundKey(State, ExpandedKey);
    for(i = 1; i < Nr; i++)
        Round(State, ExpandedKey + Nb * i);
    FinalRound(State, ExpandedKey + Nb * Nr);
}
```

Figure 4.6. *The Rijndael algorithm*

```
Round(State, RoundKey)
{
    SubBytes(State);
    ShiftRows(State);
    MixColumns(State);
    AddRoundKey(State, RoundKey);
}
```

Figure 4.7. *The Rijndael* `Round` *function*

```
FinalRound(State, RoundKey)
{
    SubBytes(State);
    ShiftRows(State);
    AddRoundKey(State, RoundKey);
}
```

Figure 4.8. *The Rijndael* `FinalRound` *function*

If the key size is larger, that is, $\mathrm{Nk} > 6$, an additional byte substitution is done if $i \bmod \mathrm{Nk} = 4$. In this case the expansion is formed as

$$W_i = \begin{cases} \mathrm{SubBytes}(\mathrm{RotByte}(W_{i-1})) \oplus R_{i/\mathrm{Nk}} \oplus W_{i-\mathrm{Nk}} & , i \bmod \mathrm{Nk} = 0 \\ \mathrm{SubBytes}(W_i) \oplus W_{i-\mathrm{Nk}} & , i \bmod \mathrm{Nk} = 4 \\ W_i \oplus W_{i-\mathrm{Nk}} & , \mathrm{else} \end{cases} .$$

Implementation Efficiency

When the contest for the design on the AES system was first announced it included directives for security, speed of encryption and decryption, memory requirements and hardware implementability. The ten finalists were thoroughly tested and analysed on many different platforms. It turned out that all of them held up to the high security demands stated. Rijndael turned out to be very efficient in both encryption and decryption. For 32-bit based platforms with few restrictions on memory usage the algorithm can be implemented very efficiently using table lookups for a lot of the operations. The other operations involve mostly exclusive or operations which also are very efficiently done.

On smaller systems the memory requirements might be different and there exists some trade offs between memory usage and processing time. For example, there is no need to precompute and hold the complete expanded key if memory is scarce. Only one Round Key is needed to be held in memory since the next can be calculated from it and replace each word as it goes on.

4.2.4 Stream Ciphers

The previous mentioned ciphers work on blocks of input and are therefore called block ciphers. Stream ciphers, on the other hand, calculate a stream of keys depending on some shared secret initial key. The key stream, z_i, is used to encrypt the plaintext stream, x_i, into the ciphertext stream, y_i, according to the rule

$$y_i = e_{z_i}(x_i).$$

Each character can be either a block of bytes – like in the DES feedback modes, a byte, or even a single bit.

The calculation of z_i is dependent on the key and the previous $i - 1$ plaintext characters. However, variants of stream ciphers that are independent of the plaintext are called synchronous. A periodic stream cipher is a stream cipher where the key stream is starting to repeat. The longer this period is the safer the stream cipher can be. A short period results in possibility of attacks against the cipher using for example the Kasiski test or index of coincidence as are described in [16, 33].

Another way to generate the key stream is to use something called a linear feedback shift register (LFSR). It basically consists of a hardware shift register where some of the bits are tapped out and combined forming the new input value for the next shift. These shift registers are efficiently implemented in hardware and

has been the basis for many military cryptosystems [29]. The shift register is called m-stage after its bit-width. Mathematically this is basically a linear recurrence relation of degree m such as:

$$z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j} \pmod 2$$

The constants, or coefficients, determine, in the hardware case, which bits are fed back into the shift register. Carefully chosen coefficients will give rise to a period of the LFSR as long as possible, that is, $2^m - 1$ [29, 33].

Since the encryption and decryption process is very efficient in hardware and also not as complex to implement as many block ciphers they fit neatly in securing stream of data over different data links. However, as the name imply they are linear and therefore vulnerable to certain attacks. This can be solved by using a non-linear feedback function instead of the simple one described above. Also, many LFSR:s can be connected through a non-linear filter before the actual output key stream is used. There are even others that are based on using one LFSR to clock another forming more non-linearity.

The connection polynomial, the coefficients in the feedback, can either be known or secret. Using known constants the key is the initial state of the shift registers. This is not as secure as using secret coefficients where these are also part of the key. Implementation wise the secret variant is more secure but will need a more complex circuitry. This can, to some degree, be compensated for by using a sparse polynomial. A sparse polynomial is, contrary to a dense one, using only a few bits in the feedback function. The fewer the bits the easier it is to analyse and break the LFSR polynomial [16, 29].

4.3 Asymmetrical Cryptography

The main problem of using the previously mentioned cryptographic systems has been that all participants must know the secret key to be able to communicate efficiently. In 1976 Diffie and Hellman came up with the idea of public-key systems. Rivest, Shamir, and Adleman realised the first system using the ideas of a public-key 1977. Their system became known as the RSA Cryptosystem which will be addressed in section 4.3.1.

There are other public-key systems that are based on the same ideas but exploit different mathematical problems to achieve the resulting system. Some of them will be described later on.

The basic idea behind the public-key cryptosystems is the trapdoor one-way function and the fact that the key is a pair of keys. The key pair consists of one public key that can be spread to everyone and one that is secret to each individual. The trapdoor one-way function is a mathematical problem that states that it should be easy to compute the public encryption but computationally hard to do the in-

verse, that is, decrypt, unless the secret key is known. This secret key function as the trapdoor.

Given a key pair (K_p, K_s), where K_p is the public key and K_s is the secret key, Alice can get a hold of Bob's public key from some public directory. She can then use it to encrypt a plaintext, x, and send it to Bob. Only Bob, who knows the secret key, can decrypt the message.

$$\begin{aligned} E_{K_p}(x) &\rightarrow \quad y \\ D_{K_s}(y) &\rightarrow \quad x \end{aligned}$$

The key pairs can also be used in the other way around to sign messages. In this way Alice can encrypt a message using her own secret key. Anyone that has her public key, or can get it, can decrypt the message. This way there is no confidentiality of the information, but everyone knows who actually sent the message. This way of using the system results in a so called digital signature. The both cases can be combined to provide for an even higher safety.

4.3.1 RSA

The RSA Cryptosystem is based on the one-way function $f : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$:

$$f(x) = x^b \bmod n,$$

where n is a product of two distinct, large primes. Further, b is the public part of the key together with n. The private key consists of p, q, and a. The relationship between a and b is:

$$ab \equiv 1 \pmod{\phi(n)}.$$

The encryption function is the f function shown above. The decryption is done using:

$$d_K(y) = y^a \bmod n.$$

The encryption function forms a one-way function since without the knowledge of the value of the $\phi(n)$ or a it is infeasible to compute the decryption function. To be able to compute a one must know which primes the n is made up of. If this information is available it is easy to compute since $\phi(n) = (p-1)(q-1)$ according to elementary number theory, see for example [12]. Breaking this system without such knowledge thus comes down to factorisation into large prime factors which still is considered quite hard.

Computational Complexity

The encryption and decryption is rather straight forward on paper. However, to get any security the system must withstand factorisation attacks which by now can factorise numbers over 100 digits long. Many current implementations uses a modulo of 512 bits which correspond to a decimal number of around 154 digits.

This is getting really close to what is possible with current factorisation algorithms and should not be considered very safe any more.

So, why not just use more bits in the modulo? Well, this is of course the only solution to this problem but at the price of computational complexity. Since the algorithm uses arithmetic operands larger than the word size of most processors we have to introduce algorithm for multi precision algorithms. The implementation details can be found in many textbooks, for example [16].

The total complexity for the encryption function, and also the decryption, is dependent on the host computers word size. Adding two multi precision numbers of k-bits using standard "grade-school" arithmetic can be done in $\mathcal{O}(k)$ and multiplication in $\mathcal{O}(k^2)$. The modulo operation reducing a $2k$-bit number to k-bits can be achieved using long division and keeping the remainder. This operation will take $\mathcal{O}(k^2)$. To get to the more essential parts the operation $xy \bmod n$ can be done by first doing the multiplication and then the reduction. These two steps can be achieved in $\mathcal{O}(k^2)$ and is called modular multiplication.

The modular exponentiation, the goal of this discussion, can be implemented using a number of modular multiplications but this is very inefficient since the exponent is very large. Instead there is a well-known algorithm called "square-and-multiply" which reduce the number of modular multiplications required to at most 2ℓ where ℓ is the number of bits in the binary representation of the exponent. Since $\ell \leq k$ the operation is done in $\mathcal{O}(k^3)$. The "square-and-multiply" algorithm can be found all over and in [16, 29].

The implementation of RSA in hardware is about 1500 times slower than a hardware implementation of DES [33]. To this we should add that the initial p and q must be found which will take some time of its own but of course is only needed to be done a few times depending on the security needs.

4.3.2 ElGamal

The RSA Cryptosystem was based on the difficulty of factoring large integers. The ElGamal Cryptosystem is instead based on the difficulty of the discrete logarithm problem in finite fields. The discrete logarithm problem is concerned about solving

$$\alpha^a \equiv \beta \pmod{p},$$

where p is prime, $a \in \mathbb{Z}_p$, and $\beta \in {\mathbb{Z}_p}^*$. The notation of $\mathbb{Z}_p$ is the equivalent of the infinite $\mathbb{Z}$ but limited to p elements. The starred form is the multiplicative group of $\mathbb{Z}_n$ which contains all elements of $\mathbb{Z}_n$ that has one (1) as the greatest common divisor with n. More formally,

$${\mathbb{Z}_n}^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}.$$

With n as a prime as above the multiplicative group will contain all elements of $\mathbb{Z}_n$ except 0 and n itself. With this short explanation of some fundamental number theoretical definitions as can be found in [12] we continue our exploration of the ElGamal Cryptosystem.

As we saw in the RSA Cryptosystem computing exponentials can be done relatively efficient using the "square-and-multiply" method. However, the inverse operation of finding the logarithm is considered a hard problem if the prime number is chosen carefully. Also, to be secure the prime number should consist of at least 150 decimal digits and $p-1$ should consist of at least one large prime. Using these assumptions and facts the exponentiation can be seen as a one-way function.

The public part of the key pair is the chosen prime number p, $\alpha \in {\mathbb{Z}_p}^*$, and $\beta \equiv \alpha^a \pmod{p}$ where a is the secret part of the key pair. The plaintext is coded as elements in ${\mathbb{Z}_p}^*$ and the ciphertext is pairs of such elements. Also, the encryption is done using a secret random number $k \in \mathbb{Z}_{p-1}$. This secret random number is generated by the encrypter and makes it possible for a given plaintext to have different ciphertexts depending on the chosen k. The encryption function is defined as

$$e_K(x,k) = (y_1, y_2),$$

where

$$y_1 = \alpha^k \bmod p$$

and

$$y_2 = x\beta^k \bmod p.$$

To decrypt this back into plaintext the decryption function is

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p.$$

As we can see in the encryption function the chosen random number is sent along the actual ciphertext as an exponentiation of α. Using this information the rightful receiver, who knows the secret exponent a, can compute β^k from α^k. When β^k is known it can be "divided" from y_2 to finally get to the actual message.

The operations involved are mainly modular exponentiations and modular multiplications and one "modular division." The complexity of these operations is $\mathcal{O}(k^3)$ where k is the number of bits of the arithmetic operations. One drawback of the ElGamal Cryptosystem is of course the extra data to be transmitted. The message expansion factor is two since each plaintext element results in two ciphertext elements [33].

4.3.3 Elliptic Curve Cryptosystems

The ElGamal Cryptosystem can actually be generalised to work on any group where the discrete logarithm problem is hard to solve. One such group is an elliptic curve. In this setting each plaintext element is mapped onto an element on the elliptic curve that is defined as the points on a curve in $\mathbb{Z}_p$ for example the elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{Z}_p$ where p is prime and $p > 3$ is the set of solutions $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ to the congruence

$$y^2 \equiv x^3 + ax + b \pmod{p}.$$

The constants a and b are chosen such that $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$. There is also a special point $\mathcal{O}$ called the point at infinity [33]. The mathematical field of elliptic curves have been thoroughly researched and more information about the mathematical details can be found in for example [33] but preferably in other literature.

The problem of using the ElGamal Cryptosystem in the setting of an elliptic curve is that each plaintext element must be mapped onto an element of the elliptic curve. This by itself makes the plaintext alphabet twice as large. If we on top of this remember that the ElGamal system itself adds a message expansion factor of two the total is a four times larger data set. In addition to this problem, the actual number of elements that exists on the elliptic curve is less than one could expect therefore introducing a harder mapping problem and smaller alphabet. Due to Menezes and Vanstone a cryptosystem using elliptic curves but overcoming these problems exists. Their system can take any pair $(a, b) \in \mathbb{Z}_p \times \mathbb{Z}_p$, even those not on the elliptic curve and mask it using operations on an elliptic curve. Also, they overcome the problem of the message expansion factor and the solution is therefore not as bad as the initial idea of using the ElGamal system.

The use of the discrete logarithm problem in elliptic curves is considered even harder than the integer factorisation problem. This leads to the ability to use smaller modulus calculation while keeping the same security level. However, there are no proofs that these problems are in fact intractable but since many mathematicians and computer scientists have studied the problems trying to find efficient algorithms to solve them have failed the common idea is that they are really hard [33].

The ability to use smaller modulus while doing the computations allows for smaller keys and faster calculations. Also, the memory resources needed are limited. Therefore the elliptic curve cryptosystems might be a good candidate for public-key cryptosystems in small mobile devices [18].

4.3.4 Comparison to Symmetrical Cryptography

So why would we ever need the “old” symmetrical cryptosystems when the public-key systems are so much better in the key handling situation. There is no need for every communicating pair to share a secret key since everybody can get hold of the public key and use it in private communications with the intended receiver. There are however some problems and the most obvious is that of efficiency. Even though there exist quite efficient implementations of the public-key cryptosystems they have no throughput to talk about in comparison to symmetrical cryptosystems. For example a high-end hardware RSA implementation is about 1500 times slower than a hardware DES implementation [18].

Also, the idea of the public-key systems is new and not as thoroughly studied as is the case for symmetrical systems. On the other side they build on mathematical problems that have themselves been exhaustively researched. Yet another aspect is of course that the previous mathematical studies have not been done with this spe-

cific application in mind and the area is therefore still immature. It is also shown that in the case of public-key cryptosystems there exist known algorithms for breaking the systems, even though not efficient but more so than brute force since the key space is so large. There is always an uncertainty that some breakthrough in cryptanalysis might lead to faster cryptanalysis rendering the cryptosystems insecure. This is of course a problem for all kinds of cryptosystems.

4.4 One-Way Hash Functions

A hash function is a mathematical function that maps a large domain into a smaller range. Hash functions are used in many different areas and the functions have different properties. Here the focus is on cryptographic hash functions. Hash functions with cryptographic properties are mainly used for data integrity checks and message authentication.

Given a message as input a hash function returns a hash-code. This value, commonly called the hash-value or hash, is not unique to this very message but has the properties that from a given hash it is hard to find the message that was used as input to the function. Albeit still hard to solve, it is easier to find a message that gives the same hash. This is something that is called the birthday attack. The attack has gotten its name from the idea of probabilities that two random people are born on the same date. Given a specific date it is very low probability in a finite population to find two people on random that match the criteria. However, given two random people the probability is much higher that they are born on the same date.

Given these properties a hash-value can be used to detect modification of messages sent over some channel. This has lead to the use for hash-values as modification detection codes (MDC). The hash-value can be sent along the message and the receiver can perform the same hash computation and check that the two values are the same. The MDC does not provide for any authentication or secrecy but the recipient can be pretty sure the message was not changed. For an intruder-in-the-middle it is thus easy to remove the original hash, change the message for its liking, and compute a new hash before forwarding the message. This is undetected by the final recipient and does pose an undesirable problem.

To solve the problem of only having modification detection it is desirable to include some better guarantee. To accomplish this, authentication is needed. By introducing a secret key into the hash function the recipient can verify not only the integrity of the message but also who sent it. This adds the need for a commonly shared secret key. Using a keyed hash function results in a hash value used as a message authentication code (MAC). Using a MAC instead of just MDC effectively disables the intruder-in-the-middle attack.

Another way is to use cryptographic signatures. Usually these are more computationally demanding why the use of hash functions still play an important role. By signing only the MDC instead of the whole message it is more computationally

$$
\begin{aligned}
f(u,v,w) &= u \wedge v \vee \bar{u} \wedge w \\
g(u,v,w) &= u \wedge v \vee u \wedge w \vee v \wedge w \\
h(u,v,w) &= u \oplus v \oplus w
\end{aligned}
$$

Figure 4.9. *MD4 functions. The operations are bitwise in MD4.* $\wedge$ *denotes bitwise AND,* $\vee$ *is the bitwise inclusive-OR operator, and* $\oplus$ *corresponds to the bitwise exclusive-OR. The* $\bar{x}$ *is the bitwise complement of* x.

efficient. The signature might be any symmetric or asymmetric signature scheme that the two peers have mutually decided upon.

The use of keyed hash functions is not as common as unkeyed hash functions for certain operations. Keyed hash functions are mainly built using block ciphers in CBC, cipher-block-chaining, mode why a special description of these are left out from this section. Instead two commonly used unkeyed hash functions are described. They both inherit design from MD4. First MD4 and MD5 are described followed by an overview description of Secure Hash Algorithm.

4.4.1 Message Digest Algorithm

The Message Digest algorithm version 5 (MD5) is based upon MD4 [16]. The algorithms are created by Ronald Rivest and described with reference implementations in [25, 26]. Both algorithms consist of a number of rounds that apply different bitwise functions to get to the resulting output. MD4 uses three rounds with each consisting of 16 steps. In MD5 a fourth round is added.

The output from both algorithms is a 128-bit hash value. As input a bitstring of arbitrary length is accepted. Since the algorithms work with blocks of 512 bits the input must be padded. In addition, to overcome the security problem of finding different length messages with the same hash a binary length value is concatenated to the message.

This preprocessing works as follows. Given the original input bitstring, append a single '1'-bit and then enough '0'-bits to get a bit length that is a multiple of 512 minus 64 bits. These last 64 bits are filled with the binary representation of the original message length modulus 2^{64}.

The resulting, formatted input is divided into m 512-bit blocks. Each block is processed in turn by applying a number of rounds and steps. These operations affect internally held variables called chaining variables which are kept between each round and input block. After the last block has been processed and the chaining variables has been updated they are concatenated, forming the output hash value.

The processing of each block is done by dividing the input block into 16 32-bit words, $X[j], 0 \leq j < 16$. These words are then processed in a number of steps in each round. In MD4 there are three rounds each with 16 steps. For each round a different bitwise function is used. The functions are shown in figure 4.9.

Initially, for each block, a set of working variables are set up from the chaining

$$g(u,v,w) = u \wedge w \vee v \wedge \bar{w}$$
$$k(u,v,w) = v \oplus (u \vee \bar{w})$$

Figure 4.10. *MD5 functions*

variables.

$$(A,B,C,D) \leftarrow (H_1,H_2,H_3,H_4)$$

Before the first block is processed the chaining variables are initialised with specified constants. In each step the working variables are updated as follows in MD4:

$$t \leftarrow (A + f(B,C,D) + X[z[j]] + y[j])\,,$$

$$(A,B,C,D) \leftarrow (D, t \hookleftarrow s[j], B, C).$$

The $+$ operator is integer addition modulo 2^{32}. The three arrays y, z, and s are helpers and means:

z An array holding which word to use from the current input block. The array holds a number between 0 and 15 in each of its elements. The actual values are predefined.

y An additive constant. In MD4 the constant is `0x00` for round 1, `0x5a827999` for round 2, and `0x6ed9eba1` for the last round. In MD5, on the other hand, a different constant is used in each step. The 64 constants are defined as the first 32 bits of binary value $4294967296 \times \mathrm{abs}(\sin(j+1))$, where $0 \leq j < 64$ is in radians.

s Similar to the z array the s array holds the number of bit positions to left shift (cycle).

The subscript j for the first round is $0 - 15$, the second round $16 - 31$, and for the third round $32 - 47$. Additionally, MD5 adds a fourth round, that is $48 \leq j < 64$.

In each round a different bitwise operation is used. These operations are defined by the functions f, g, and h shown in figure 4.9 and are used in round one, two, and three respectively. In version 5 of the MD algorithm one additional function, k, is defined for the fourth round and the second round function is redefined as is depicted in figure 4.10.

In MD5 each step of processing is changed from updating working variable B as a left shift of the temporary variable t to also include the previous value of B. That is, change the updating of B from $B \leftarrow (t \hookleftarrow s[j])$ to $B \leftarrow B + (t \hookleftarrow s[j])$

After processing each 512-bit block the chaining variables are updated from the working variables,

$$(H_1,H_2,H_3,H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D).$$

After all the input have been processed the hash value is the concatenation of the chaining variables as: $H_1||H_2||H_3||H4$.

4.4.2 Secure Hash Algorithm

In this section the Secure Hash Algorithm (SHA) in revised form (SHA-1) is described. SHA-1 is based upon the MD4 algorithm. However, since the output hash value is larger, 160 bits as opposed to 128 in MD4 and MD5, it is more secure against brute force attacks. The internal processing, as shown below, also provides for added security thanks to the redundancy of input words used in an expansion step.

The SHA-1 algorithm has many design similarities to MD4. Since the output hash value is larger a fifth chaining variable is used. To initialise this variable another constant is defined in addition to the four defined in MD4 and MD5. The SHA-1 algorithm also uses per-round additive constants as in MD4 but uses four rounds each consisting of 20 steps. The arrays with bit shift values and word access order is not used in SHA-1.

The initial preprocessing is similar to MD4 and MD5 with some byte order differences in the length field. After initialising the chaining variable each input block is processed by setting up some working variables as in MD4 and MD5. Each block of 16 32-bit words is expanded into 80 32-bit words with the first 16 corresponding to the original input and $X_j \leftarrow ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \hookleftarrow 1), 16 \leq j < 80$, and X_j denoting $X[j]$. The processing step is as follows:

$$t \leftarrow ((A \hookleftarrow 5) + f(B, C, D) + E + X_j + y_{round}$$

$$(A, B, C, D, E) \leftarrow (t, A, B \hookleftarrow 30, C, D)$$

where the MD4 function f is used in the first round, h is used in the second and fourth round, and g is used in the third round. After the processing of each block the chaining variables are updates as in MD4 and MD5 with the fifth variable following the same pattern, that is,

$$(H_1, H_2, H_3, H_4, H_5) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D, H_5 + E).$$

The completion of the algorithm is also similar to MD4 and MD5 but, as in the padding step, the byte order is specified to be the opposite.

4.5 Public Key Infrastructure

As we saw previously the public-key cryptosystems solves one of the biggest problems with key management by allowing the public part of the key-pair to be available to everyone. This cuts down the key management overhead enormously. There are, however, still some problems with this approach. If there is an eavesdropper he or she can easily see the public key, but since this is public and no security rests in that this is not to be considered a security hole. On the other side, if the attacker decides to fake a public key saying it belongs to someone else how can we then be sure of whom the public key really belongs to?

Subject	Distinguished name, public key
Issuer	Distinguished name, signature
Period of validity	Not before date, Not after date
Administrative information	Version, Serial number
Extended information	

Figure 4.11. *X.509 Certificate format*

To solve this kind of situation the use of certificates is one way to go. Each public key is tagged with whom it belongs to and this information is guaranteed by a certificate. The use of certificates leads to yet another question. How can we know that the certificates are real and not faked? By the use of trusted certificate authorities this can be the case. If we have a central certificate authority (CA) that holds the public keys together with the real identity of the parties they belong to and can certify that they are really correct the problem is solved. The CA can use a public-key cryptosystem to sign the information and each client holds the CA:s public key from factory setup, a so called "root key." This way the client can authenticate that the public key it gets from the CA is really for the party that is intended. This leads to a trust model that means that every client must trust the single CA.

The most used format for certificates is the X.509 standard. This states the fields a certificate holds and what they mean. The fields are described in figure 4.11. The X.509 certificate standard is part of the global X.500 directory standard. However, this standard will most probably never be the public, global directory it was intended to be. The naming issues are one problem of this. The X.500 naming standard uses locally unique names, but put into the directory hierarchy they can be unique globally. Another issue is that most companies are not that happy to give out a public list of employees, customers, or partners since these data can be seen as confidential in most businesses. Even though the certificate standard was thought of a part of this globally available structure the certificates can be used independently.

It turns out that the hierarchical naming problem leads to a longer chain of certifications. To address this issue Simple Public-Key Infrastructure (SPKI) was proposed in [5]. The theory in SPKI allows for logical inference for deduction of certificates ($A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$) creating chains of certificates. By following such links backwards an entity can conclude the validity of a leaf certificate. It is also possible to cache the computations for some links so that they are available later on.

The security and trust lies in the CA's ability to keep its private keys safe and to be able to check the identity of the parties for which it stores the public keys. That is one of the big problems with CAs. Before it can sign a public key it must make sure it really is the rightful "owner" that has created the public key and not someone else trying to impersonate some party. The client that uses a CA must

trust that the CA has done this in a correct way [4].

To distribute the trust from one single CA a web of trust can be woven like in Pretty Good Privacy (PGP). In this weave there exists no single CA. Instead different parties meet and sign each other's public keys. When one would like to communicate with another the public key is fetched and verified against the signatures on the public key. If the public key is signed by oneself or by someone that is trusted the key itself is trusted. This chain, or weave, grows as more and more users are taking part in signing their friends' keys. There is, of course, no real integrity guaranteed unless such an authentication chain can be found. Also, one needs to trust not only one other party but any number depending on the length of the chain. This is part of the trust model. Instead of trusting one authority we divide the trust into many parties each with a lower trustworthiness. It enough such relations exists it can be concluded that the public key really belongs to a certain party.

Also, all the security in both kind of system lies in the secrecy of the secret part of the key. If this key is leaked out then the public key can no longer be used since other parties can read encrypted data and sign as the rightful owner even though they are not. On the other side the verifying party must trust the CAs and the "root-keys" that it holds. If someone is able to insert yet another "root-key" the authentication is no longer possible. This implies that both the holder of a secret key and the verifier, using only public keys, must be secure for any kind of attack that can compromise these assumptions [4].

In the case of the web of trust model, we have yet another integrity problem. From the authentication chain it can be possible to see who someone's friends or colleagues are, information that might be considered breaking integrity by some people. This is probably not that much of a problem since other kind of traffic analysis also might get this information anyway.

4.6 Hybrid Cryptography

To overcome the key management nightmare of symmetrical cryptosystems and the problem of computational inefficiency in the asymmetrical case a hybrid variant can be used. In this way it is possible to get the best from two worlds. One way to achieve this is to use public-key systems to exchange the secret keys needed and then switch over to a symmetrical cryptosystem ones the keys has secretly been exchanged for the bulk of the data transfer.

In this approach a small amount of data is encrypted using the public-key system and then the bulk of information is sent using the more efficient secret-key system. Secure Socket Layer (SSL) is an implementation that uses this approach. It was developed by Netscape and released for the public to use. It is now part of most web browsers and web servers around the world and is the *de facto* standard. It is however sensitive to an intruder-in-the-middle attack at the initial stage when the public-key is exchanged. To overcome this it has to depend on a trusted third

party as a CA that both parties must trust. This could be a problem as noted above.

One advantage of SSL is its support for a multiple kinds of cryptographic and compression algorithms. Since a widespread use implies heterogeneity among clients the need to negotiate what protocols to use is mandatory. This is the real strength that has made SSL the *de facto* standard it is today. A more evolved and standardised method is the Transport Layer Security (TLS) which in large parts builds on SSL [1].

The SSL protocol starts of by negotiate what protocols to use in the different stages of the communication. This is all done in clear text. After this an optional stage of certificate exchange takes place. Here one or both parties sends a certificate containing their identity, public key and other information signed by some trusted third party. After the initial negotiations and certification exchanges the communication is done with the help of the public keys exchanged. This part of the session is often relatively short; it mainly consists of the exchange of secret keys to use for the coming stages of the session. When this is done the switch to secret-key cryptography can take place. Any time during the secure session a new negotiation of protocols can take place allowing for changing keys or protocols.

The approach of a hybrid system is only favourable when a relatively large amount of data is to be transmitted. This is relative to the key size that is needed to be transmitted using the public-key system initially. If only small amounts of data are to be exchanged it can certainly be done directly. However, if repeated exchanges of small data packets are to be sent now and then an onetime exchange of secret keys might be preferable and then reused. There is of course a security risk with reusing the same key for much data as the probability for an eavesdropper to get a known-plaintext attack successful.

4.7 Threshold Cryptography

In the case of a central CA all the security is centralised. If this CA is compromised all the following communications are insecure. To overcome this kind of security risk it is suggested to distribute the security over a set of multiple servers. However, by just mirroring the data to multiple servers there is no gain in security other than maybe uptime and load-balancing advantages. Instead the need to share parts of the secrets is needed, and if one server is compromised the other should not be affected. A verifying party should be able to conclude the correct verification if it can get an answer for some percentage of the servers, a threshold. This is where secret sharing systems and threshold cryptography comes into play.

The definition of a threshold scheme is a scheme where t and w are integers and $t \leq w$. The (t, w)-threshold scheme is a method of sharing a key among w participants in such a way that any t participants can compute the value of the key, but no group of $t-1$ or less can do so [33]. Initially the secret must be created by a "dealer" that hands out the secret securely to each participant. No other participant should know about the part of the secret that another participant has. At a later time

a number of participants can collectively compute the shared secret if there is at least t of them. The collective computation can be done using a trusted third party that collects the secret parts and computes the shared secret; however, using this method makes the whole process of sharing the key worthless since a compromise of the third party might render the system insecure anyhow. There are of course ways to get around this problem. By using partial signatures each part signs the document and when at least t correct signatures are done the complete signature is finished without letting any one else know each share needed for the computation. Here the first variant will be presented since the ideas are similar only some more complicated math is used.

The idea of secret sharing and threshold schemes were independently formulated by Adi Shamir and George Blakley in the mid 1970:s. The Shamir threshold system is based on a LaGrange interpolating polynomial scheme. In this scheme each share is a value calculated by the "dealer" from a set of polynomials. These polynomials have the degree of t in an (t, w)-scheme. Using t such values in a linear equation system it is possible to solve for the shared secret. Using less than t values will lead to an underdetermined system and thus give no clue to the solution. All the calculations are done in $\mathbb{Z}_p$ where p is a large prime [29, 33].

Blakley's idea is based on Euclidean spaces. If the threshold scheme is $(3, 5)$ for example the shared secret is in the regular three dimensional space. The shared secret is a point in space. Each part or share is a plane containing the shared secret, or point. If enough number of planes exists, that is, we have at least t number of participants the shared secret is the point where all planes intersect. If some shares are missing we only get a plane or a line with the actual secret still undetermined. This can of course be generalised into any t-dimensional space [29].

5

Security in Ad Hoc Networks

In earlier chapter the ideas of ad hoc networks and security has been introduced on their own. Together some new problems occur which was mentioned in chapter 3. In this chapter the applied usage of cryptographic primitives in tight coupling with ad hoc networking protocols are studied.

The chapter begins with a short classification of different levels of security needed for different applications. After this introduction some proposed solutions are presented. The first problem is the key management which is discussed in section 5.2. With proper key handling the solutions for authentication is discussed further in section 5.3. Section 5.4 finally discusses some solutions to achieve security in the routing stages.

Overall end-to-end security is left for higher layers in the protocol stack as in usual Internet engineering spirit. Most solutions that is valid in standard Internet is also available at this level for ad hoc nodes except for the on line key management services. There are, as will be shown, some solutions for ad hoc networking for this. Reusing these techniques in higher layers might provide better efficiency in some applications. However, the usage is closely related to the application in hand and there exists no general solutions.

5.1 Needs for Security

The actual needed security will depend on the type of application and the value of the information that travels in the network. This is always a trade-off that must be considered for any type of security system. Is it worth the money to build or implement a secure system when there is no valuable information flowing around? For how long must the information be safe? forever or is it just tactical real time information that is worthless in a few hours or so? These question and many more are needed to be considered when making a secure system.

Some ideas of the need for security in different types of applications are given below, ranging from low to extremely high security needs. What must be added is the need for making the systems practically usable which often leads to weaknesses in the form of human beings.

5.1.1 Home Networking

For a home network connecting the TV, stereo, VCR, DVD player etc. to a remote control the need for security might not be that high. Of course, the integrity aware people might not like the idea of neighbours being able to listen in on the control traffic and conclude what shows are being recorded and so on. However, these are more of a comfort issue than a real security threat. Also, it should not be possible for a bypassing individual that bought a remote control of the same brand to control the appliances.

On the other hand, if the network is expanded to include control over the burglar alarm and carport some additional security measures must be taken. With such addition the value of the protected information, the house and all the appliances included, increases and so does the need for keeping the network safe.

5.1.2 Emergency Services

In the case of emergency services using an ad hoc network for their communication the main need is the availability. For disaster areas there is probably no real need for securing the actual traffic since the most important thing is to be able to direct and control the rescue teams.

In the case of terrorist attacks the network should also protect the integrity and be safe against radio jamming since these attacks otherwise may lead to even worse situations. To go further along these lines, for a police squad trying to get control of a hostage situation the need for them to hide their communication from the kidnapper is vital. On the other hand, some communications might also be needed to be kept from the journalists until certain investigations are over.

5.1.3 Military Applications

The highest need for security is probably within the military tactics. An ad hoc network in the battle field can be very valuable to the enemy if they can get hold of a node. This is worth a lot and therefore the attacker might not stop at any costly attack method. This implies that the security of the network must be high enough that even the most advanced and expensive attack will not jeopardise the security of the network.

5.1.4 Physical Security

The need to secure information has mostly been addressed in the context of traffic flow. In the case of small mobile devices the aspect of physical security is even higher. A stationary node can be locked in using locks and chains but this will not be a valid solution for a mobile device. This poses some extra security related problems on ad hoc nodes. Since they can not be physically protected in the same way as other equipment solutions based on tamper proofed chips are proposed.

Such solutions can provide for some safety of the data even in the case of theft of the device.

5.2 Key Management in Ad Hoc Networks

The major problem of ad hoc networks is the lack of an on line key management service. Depending on the nature of the ad hoc network and the usage different solutions might be available. In this section four such solutions are proposed. In the first proposal Stajano and Anderson gives a solution for small nodes without any need for human input or feedback. It is closely modelled on the natural phenomena of imprinting. After this, two related proposals are presented. Both are based on threshold cryptography but in some what different settings. Finally a distributed solution is presented where each node is part of the key service in a PGP like manner.

5.2.1 The Resurrecting Duckling

The Resurrecting Duckling security policy model is based on the analogy of newborn ducklings that are imprinted to the first moving thing that makes a sound, which should of course be the mother duck. The ducklings are devices that can have different states or souls as Stajano and Anderson calls it.

At first, the duckling is in an imprintable state waiting for the mother duckling to imprint it. The process of imprinting is done using a secure channel such as direct electrical contact in which a shared secret is sent in plaintext. After the duckling has been imprinted it cannot be imprinted by another device unless it "dies" or the mother duck tells it to die. The duckling can be programmed to die on certain events or time outs which may be analogous to old age. While holding the secret key it can be used to communicate or authenticate the duckling with the mother duck using any cryptographic methods that are available in the two devices and using the key. Since the shared key is a shared secret it is possible to use symmetrical cryptography if wanted to keep the data transmitted private.

This approach was described in [30] and later extended in [31] to include the ability to allow communications between siblings. This is accomplished by letting the mother duck download a security policy that can, for example, be an access control list which states what can be done on the device by whom. This greatly widens the usefulness of the idea but still gives rise to some disadvantages. Instead of using access control lists the ducklings can use certificates signed by the mother duck to trust each other. This, however, puts some additional demands on the devices such as secure clocks for the ability to have validation periods on the certificates. This is a large constrain since the problem of keeping a secure clock is quite demanding by itself. As noted in [31] there exist methods where the valid period is short enough so that the need for secure clocks in each device is not needed. This often implicates that new certificates must be downloaded equally

often adding the need for an on line CA.

The resurrecting duckling security policy model adds the ability to authenticate a node to its master and in the extension also some siblings. Also, there are lengthy discussion around tamper proofing and tamper evidencing since if a malicious attacker could get hold of the shared key it can masquerade itself as an authentic node. Since all nodes need a master there are applications in which this is not possible. In the home network application case it is, however, a well suited approach.

5.2.2 Secure Key Distribution

Khalili, Katz and Arbaugh [13] suggest an ID-based threshold reinforced cryptographic scheme overcoming the common needs for *a priori* distribution of secret keys or public certificates. The model is based on identities, which each node has, that is used as the public key. Threshold cryptographic principals are used to build a private key generation system that each node can turn to get its private key.

The use of each node's identity saves bandwidth since there is no need to distribute public keys and signatures from the CA. Also, the ID is usually much shorter than public keys used in most public key protocols and also present in all data packets any way. Furthermore, the computational overhead in creating the public/private key pair is reduced. There is no need to compute any public key. The private key must be created by some distributed nodes. This should only be needed to be done once for each new node in the network.

The benefit of using a threshold system for the private key generation (PKG) is of course the need to not be dependent on any one special CA. In addition, the mobile nature of ad hoc networks allows for some robustness using the threshold since only a number of the key servers needs to be in a connected network. If not enough servers are available, the node can simply move to another location after collecting those that are available at the previous location. This way, the node will finally have received a complete private key.

The whole scheme consists of four different parts, or algorithms.

Setup When the network is initially formed the present nodes must agree on the security parameters, the threshold scheme, and so on. The setup stage takes as input these parameters and generates the master public/private keys for the system.

Extract For each node to get its private key it needs to contact the distributed PKG service. This service extracts the node's private key using the master secret key and the node's identity as input. This step is vulnerable to some attacks since the identity might be easily spoofed.

Encrypt The encryption is done by each node using the public master key, the recipient identity, and the message plaintext. The output from this step is of course the ciphertext to send.

Decrypt At the receiving end, the node uses the master public key, the incoming ciphertext, and a personal secret key to decrypt the ciphertext into the plaintext.

To address the problem of identities being spoofed Khalili, Katz, and Arbaugh suggests using unpredictable identities chosen when the node enters the network. Additionally, the PKG service should refuse to hand out any private key to each ID more than once. This effectively solves the problem of spoofing since any adversary will be unable to guess the identity of any node in advance. Yet another solution involves statistically unique cryptographically verifiable (SUCV) addresses.

Each node in the network needs to know about the master public key to perform the encryption and decryption. The master public key is easily spread by broadcast or any other way. However, this leads to vulnerabilities against intruder-in-the-middle attacks for nodes joining the network. A malicious node can send out any key to the new node, pretending the key to be the master public key. In the role of attacker, this public key is of course the key to a PKG service which that adversary has control over, efficiently implementing an intruder-in-the-middle attack [13].

5.2.3 Distributed Key Management

As noted many times before is the problem of usage of on line trusted parties, key servers, and CAs. To address this specific problem it has been suggested that the CAs are replicated and therefore available to a larger extent. Even though it is more available it is also more vulnerable. If an adversary can compromise one CA all the secrets kept in that, and all the others, are also compromised thus weakening the security in the system at large.

One solution is that of threshold cryptography that was introduced in section 4.7 on page 46. In their paper Zhou and Haas [37] introduce a distributed key management service based on this. The service at large has a public key known to all nodes in the system. The nodes trust certificates signed by the service's private key. The service can be queried by the nodes in the system to get the other nodes' public keys. The nodes can also submit update requests to update their own public keys.

The service consists of n special nodes in a threshold $(t+1, n)$ configuration where $n \geq 3t+1$. These nodes are the servers of the service. The servers themselves also have public keys to the other servers and thus are able to construct secure channels between themselves. The threshold scheme allows for an adversary to compromise up to t servers in any period of time. However, the assumption is that the adversary lacks the computational power to break the cryptographic schemes that are used [37].

In this key management service the servers share the ability to sign certificates. When the service is requested each available server partially signs the certificate. These partial signatures are combined by any combiner node to retrieve the complete signature. In the case of compromised servers the combiner can check the

validity using the public key of the service and if it is not valid the combiner node tries another set of $t + 1$ signatures. The combiner itself need not know any secret and can thus be any node in the network.

The actual threshold scheme used can be any that fulfils the requirements set up. To be more robust the scheme employs share refreshing and allows for automatic adaptation of the configuration. Thus when non-compromised servers identify compromised one they can change the threshold scheme to exclude them. The share refreshment allows the servers to compute new shares from the old one that they already hold. In addition, since the new shares are independent of the old shares a compromised server can not use the old shares in any combination with the new shares to sign a certificate. Thereby forcing the adversary to compromise another set of $t + 1$ servers between each update [37].

The usage of threshold cryptography is very promising and fits nicely into the ideas of non-centralised services. However, there are large demands on signature validation which might not fit small nodes with limited computational power that well.

5.2.4 Self-Organised Public Key Infrastructure

One way to overcome the centralised issues in PKI using CAs was to let the users themselves provide the certificates as is done in PGP. However, PGP itself still suffers from the problem of distributing the certificates using public key-servers which are kind of centralised servers that need special treatment as such. To be really decentralised this storage must be accomplished without the need for any specially treated nodes. This is where the self-organised PKI comes in.

Using the same web of trust idea as PGP this approach puts the public certificates at each user. However, all users can not know about every certificate and every other user. Instead each user hold a small number of certificates and when needed merge these certificates with some other node together getting enough certificates to enable the verification of a public key using a trust chain in the merged set of certificates.

In their paper [10] Hubaux, Buttyán, and Čapkun describe a graph algorithm in which each user, using only locally available information, calculates a sub graph from the complete trust graph. The algorithm, which they call "The Shortcut Hunter algorithm," proves to give quite good results when running it on actual PGP trust graphs. It gives a scalable, decentralised solution to the problem of PKI certificate distribution. They showed that the probability of two nodes' merged certificate lists holds a certificate chain between the two of them was high using a relatively small number of certificates in each node. Of course, this relates to the size of the complete trust graph and the assumption of it to be connected [10].

As is the case for all such trust graphs it is possible to extract information about who knows who, or at least who trusts who. This is the foundation of the decentralised certificate authority approach but may be used in malicious ways by adversaries.

5.3 Authentication

With only a valid key without knowing who it belongs to, or an integrity check on the data packets there are no real security achieved. Without knowing who you talk to, or at least, knowing that the one you are talking to are the one he or she claims to be, there is no real security in the system.

This section is about the problem of authentication. Authentication is usually based on knowledge or trust and in the Internet this is commonly achieved using public key cryptography and digital signatures. Here two different approaches are presented. The first is based on a distributed model of trust and references, much like we act person to person in different situations. The second proposal is used to make the verification more efficient by not using public key cryptosystems as the main verification system. This second proposal fits neatly in broadcast environments and ad hoc networks where some packet might not reach the destination due to sporadic network connectivity changes.

5.3.1 Light-Weight Authentication Model

Contrary to many of the other authentication protocols discussed so far this model does not base its foundation on strict mathematics. Instead the model tries to simulate the human behaviour. In this model devices, and their users, are authenticated by the use of references.

When a node wish to communicate with another node the destination node is asked to supply common knowledge such as a shared key. If this is not available the source node checks its list of trustworthy devices and asks them if they can authenticate the identity of the target node. These nodes can do the same in recursive manner to finally come up with a ‘yes’ or ‘no’. The source node can decide if the result is good enough by using a threshold of the number of ‘yes’-answers in relation to the total number of answers.

The initiating node can also ask the destination node for a list of references. These referencing nodes can be asked if they know the destination node. Also, using the trust relations as previously each reference can be authenticated to make sure they are not setup to work on behalf of the destination node.

For the evaluation of the trust for the destination node the initiator can implement different limits as the threshold. Also, it can check each recommendation chain for suspicious nodes and ignore these chains during further evaluation. It is also possible to put different weights on different trust chains. For example, the chains of the references given by the destination might be weighted lower than the trust chains acquired through the list of already trusted nodes.

Once the authentication is finished a secure channel can be setup. One way is to send a random value over a trustworthy channel, that is, via nodes that the initiating node trusts, explicitly or implicitly. The destination node sends another random value using a random path. Using these two values the both nodes are able to compute a shared secret key. Of course, this approach is vulnerable to nodes

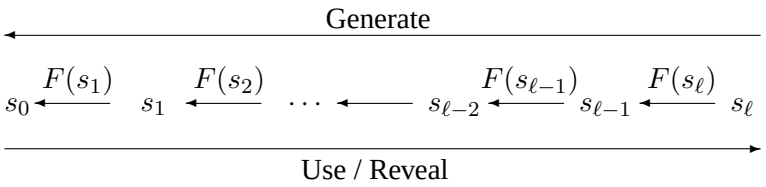

Figure 5.1. *One-way chain example [24].*

overhearing both random values.

If the nodes are able to use public key cryptography, a key agreement protocol based on this will be available for the two nodes. However, in this case the setup is vulnerable to active intruder-in-the-middle attacks.

Furthermore, Weimerskirch and Thonet suggest a feedback message that can be sent out if a node that has been authenticated has shown to be untrustworthy. The initiating node can send out this feedback to the trust chains that said that the node was trusted. It is also possible to broadcast this kind of information. Since the shared knowledge to the real node is still available the benign node that was the intended destination is not harmed by this information [35].

5.3.2 Timed Efficient Stream Loss-tolerant Authentication

A general approach to authenticate broadcast message is to use public-key systems where the sender can sign the message and all receivers verifies the signature. This leads to high overheads since signing and verifying take some time to perform. It also takes up bandwidth resources since every verifying node need to contact an on line CA to get the public certificate needed for the verification. This might not be needed every time but the security level might need a check for certificate revocation periodically. The Timed Efficient Stream Loss-tolerant Authentication (TESLA) protocol for broadcast authentication proposed by Perrig et al. [24] solves these problems.

One-Way Chains

The basic working of the TESLA protocol is based on one-way chains. A one-way chain is generated by repeatedly apply a one-way hash function. Each value can be computed using the previous in the chain. The values in the chain are used in reverse order, thereby not revealing any secret information until later on.

As shown in figure 5.1 the generation of a one-way chain is controlled by the sender. At first a random value, s_ℓ, is chosen. From this value the one-way chain is iteratively computed forming the values $s_{\ell-1}$, $s_{\ell-2}$, ..., s_1, s_0. These values can later be used in the reverse order, that is, by using s_0 first. The value s_0 is said to be a commitment to the entire chain. This means that any element s_i can be verified by applying the one-way hash function on s_i repeatedly until s_0 is reached.

More formally, to verify that s_i is the element of index i in the chain we check if $F^i(s_i) = s_0$.

Since the memory is limited, and so is time, it is not possible to create an infinite one-way hash chain. Using a long chain takes much memory and a large initial computation time. There exists a memory time trade of. By postponing some computations to later it is possible to reduce the memory needed to hold the chain. It is possible to store the chain in $\mathcal{O}(\log n)$ using $\mathcal{O}(\log n)$ time.

In TESLA the values are used as keys for signing the message to broadcast.

Time Synchronisation

The TESLA protocol is based on the assumption of loosely time synchronised nodes. That is, the receivers need to be able to approximate an upper bound on the sender's clock. This can be accomplished using a simple time synchronisation protocol if needed. Each receiver performs the following step to make their approximation of the maximum time synchronisation error. The receiver sends a time synchronisation message to the broadcast source noting the time it sends out the message, t_R. The message includes a randomly generated nonce identifying the specific message. When the server receives the message it notes its time and then creates a reply message containing the sender time, t_S; and the received nonce. Before transmission back to the node that asked for it the server signs the message using its private key. This way the receiver can verify that the time message was not altered on its way back. Using the time of reception the receiver node can calculate the time synchronisation data it needs.

TESLA Protocol

The TESLA protocol divides the upcoming time into intervals of a specified length. Each such interval is assigned one key from the one-way chain of key values. The sender also decides a disclosure time that states how many intervals that must go by before an interval key is disclosed.

During the broadcasting the sender attaches a MAC to each message it sends out. The key for these MACs is the one for the current time interval. With each message the key from the previously disclosed interval is attached. When a receiver sees such a message it stores it waiting for the key to be disclosed. If the message is delayed in some way and the receiver know that the key has already been disclosed it can no longer trust the packet. However, if the receiver can know that the secret key has not yet been disclosed, as it can since it knows the key disclosure schedule, the packet is buffered. When a few intervals has elapsed and the interval key for a previous interval has been revealed the receiver can verify that the message was really authentic.

The server does not use the actual interval key to sign the messages. Instead, a signing key is generated from the interval key by applying some known one-way

function. This way the key is only used for one thing and not both signing and generating the one-way chain [24].

Since the TESLA protocol only uses symmetric cryptographic primitives it can be implemented very efficiently. The protocol can also be used in a setting working as a PKI solution. However, initially the time synchronisation does need a way to securely exchange data before the TESLA protocol can work.

5.4 Secure Routing

5.4.1 Secure Routing Protocol

The Secure Routing Protocol (SRP) proposed by Papadimitratos and Haas [19] is mainly focused on the route discovery process. The routing protocols mentioned earlier in section 2.3.2 were not concerned at all with security and there are many attacks available to disrupt the network connectivity by targeting the routing protocol.

The SRP uses symmetric cryptography which is efficient for message authentication codes between the peer nodes in a communication channel. The intermediate nodes need not do anything but pass on the routing request and data traffic. This does, however, imply the availability of shared secrets between every two nodes that are to communicate with each other. The problem of key distribution is not addressed in [19] other than using key negotiation algorithms such as for example the Elliptic Curve Diffie-Hellman algorithm.

SRP can be added to a number of different routing protocols quite easily but in [19] Papadimitratos and Haas describes the setting using a DSR like protocol as base. The basic assumption is the knowledge of a shared key that can be used initially for authentication of the peer nodes and the correct routing information. Later on during the actual information exchange the keys can be used for integrity and privacy of the data, thus creating a secure channel.

Many reactive routing protocols, including DSR, DSDV, and AODV uses monotonically increasing sequence numbers to identify requests and suppress duplicates flooding the network. The use of these sequence numbers enables attacks such as the rushing attack, see rushing attack in section 3.2.2 on page 19. The need for this ability is great and even SRP uses it. In addition a random request identifier is also added to the message. This way the rushing attack has a low probability of succeeding since the identifier used are hard to foresee. The use of this random number does involve the use of a secure pseudo random number generator in each node initiating a route request.

When the source node creates the route request message the non-mutable fields of the IP packet, the source address, the sequence number, the random identifier, and the shared key are all input to a hashing algorithm. The hash is also part of the SRP header. The request is flooded out, as is done in most reactive protocols, to all neighbours. Each neighbour adds itself to the list of nodes on the path and forwards the message. There is no need for each intermediate node to do any cryptographic

or otherwise computationally expensive operation. When the target node receives the route request it verifies that the request indeed comes from the correct source, a source with which it shares a secret key. It then constructs a route reply and creates the MAC using the complete path information as input to the hash algorithm. It then sends the route reply back in the reverse path it came on. One strength of ad hoc networks is the redundancy which is taken advantage of in SRP. The target will receive multiple route requests from different routes. It can reply to each of these giving the source node the ability to choose one of the many alternatives. The target will have to limit the number of replies so that a malicious node can not consume too much bandwidth or resources by forwarding spoofed route requests.

When the source node receives the route replies it can validate the correctness of the route by verifying the MAC. Since the reply was routed using the path information in the reply packet the source node can be certain that the route was actually followed and correct. So far so good, to ensure communication with the target the source can choose one or more of the routes that did get back. Even if an adversary drops all data traffic on a given route another route can be chosen. This works as long as there are enough benign nodes in the network. To further take advantage of the redundancy the ability to use diversity coding can make the communications even more reliable. In diversity coding the techniques of error correcting codes and redundant information is used. The actual data is split up and sent along multiple different routes. When the target receives the pieces it can combine them to the complete message even though not all of the pieces arrived correctly or at all [19, 37].

5.4.2 Ariadne

Ariadne is a reactive secure ad hoc network routing protocol based on DSR. DSR was conceived by Hu, Perrig, and Johnson. It is mainly aimed at using TESLA as the authentication protocol. In the route discovery Ariadne includes the source and destination addresses, a request ID and a time interval used for TESLA. In the packet header a field for a hash chain is included and also two lists, one is the node list as in the original DSR and the other is a MAC list. The hash chain is initialised by the initiator of the request by making a secure MAC using a symmetric secret key. The input for the MAC calculation is the source, destination, request id, and the time interval value. The request is broadcast to all the initiators neighbours.

When a node receives a route request that is not destined for itself it calculates the next hash chain value using a one-way hash function with its own address and the previous, incoming, hash chain value. It also calculates the next field in the MAC list using its TESLA key for the specified time interval. Before forwarding the request message the fields are updated with the newly computed information. The hash chain value is replaced while the address and MAC values are appended to their respective lists.

The target of the request checks the validity of the request by verifying that the intermediate nodes' TESLA keys have not been disclosed for the time interval yet.

$$
\begin{array}{ll}
S: & h_0 = \text{MAC}_{K_{SD}}(\text{REQUEST}, S, D, id, ti) \\
S \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, h_0, (), () \rangle \\
 & \\
A: & h_1 = H[A, h_0] \\
 & M_A = \text{MAC}_{K_{A_{ti}}}(\text{REQUEST}, S, D, id, ti, h_1, (A), ()) \\
A \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, \underline{h_1}, (\underline{A}), (\underline{M_A}) \rangle \\
 & \\
B: & h_2 = H[B, h_1] \\
 & M_B = \text{MAC}_{K_{B_{ti}}}(\text{REQUEST}, S, D, id, ti, h_2, (A, B), (M_A)) \\
B \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, \underline{h_2}, (A, \underline{B}), (M_A, \underline{M_B}) \rangle \\
 & \\
C: & h_3 = H[C, h_2] \\
 & M_C = \text{MAC}_{K_{C_{ti}}}(\text{REQUEST}, S, D, id, ti, h_3, (A, B, C), (M_A, M_B)) \\
C \rightarrow *: & \langle \text{REQUEST}, S, D, id, ti, \underline{h_3}, (A, B, \underline{C}), (M_A, M_B, \underline{M_C}) \rangle \\
 & \\
D: & M_D = \text{MAC}_{K_{DS}}(\text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C)) \\
D \rightarrow C: & \langle \underline{\text{REPLY}}, D, S, ti, (A, B, C), (M_A, M_B, M_C), \underline{M_D}, () \rangle \\
 & \\
C \rightarrow B: & \langle \text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (\underline{K_{C_{ti}}}) \rangle \\
 & \\
B \rightarrow A: & \langle \text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, \underline{K_{B_{ti}}}) \rangle \\
 & \\
A \rightarrow S: & \langle \text{REPLY}, D, S, ti, (A, B, C), (M_A, M_B, M_C), M_D, (K_{C_{ti}}, K_{B_{ti}}, \underline{K_{A_{ti}}}) \rangle
\end{array}
$$

Figure 5.2. *Route discovery example in Ariadne. For each step updated fields are underlined to ease reading. The initiator S wants to find a route to the destination D [9].*

It also makes certain that the hash chain is correct by computing the whole chain; since it shares the secret key with the initiator it can do this. If these checks certify that the request was valid a route reply is constructed and sent back to the initiator. The reply packet consists of the target and initiator addresses, time interval, node list, MAC list, target MAC, and a key list. All the fields prior to target MAC correspond to the fields in the incoming request. The target MAC is computed by the target on these fields using the shared key that it shares with the initiator. For each intermediate node on the return path the nodes add their disclosed key that was used to create the MAC they appended to the MAC list as the request went towards the destination.

Using the key list and the target MAC the initiator can verify the route and conclude that there were no forgeries of routing data. An example of a route discovery is shown in figure 5.2.

To maintain a previously discovered route the approach is based on the DSR protocol. As in DSR an intermediate node returns a route error packet back to the initiator if it detects that it can no longer send packets along the specified path. To secure this approach so that adversaries can not send forged error packets the Ariadne protocol demands that the error packets needs to be authenticated. When using TESLA the route error packet consists of the sending and receiving nodes' addresses, the expected destination time interval, an error MAC, and the recent TESLA key. The sending address is the address of the intermediate node that

sends the error message while the receiving address is the node that couldn't be reached. The error MAC is calculated over the address fields and the time interval. The recent TESLA key is the most recent key that the node has disclosed [9].

Performance Evaluation

Simulation studies of Ariadne, DSR, and DSR without any optimisations have been made by Hu, Perrig, and Johnson [9]. Their results show that the regular, optimised DSR outperform the other two but the study also shows what the effect of adding security awareness has on the routing protocol. For comparison the three protocols have been simulated with the same setup. The results are presented as six different metrics.

Packet Delivery Ratio This is the ratio between sent packets and packets that are actually received by the intended recipient.

The optimised DSR show a very high ratio above 0.97 in all settings. When it comes to Ariadne and unoptimised DSR the results vary more drastically. For highly mobile nodes the unoptimised DSR has better performance but as the nodes are getting more stable the delivery ratio is higher for Ariadne. The authors conclude that this is because of the slight delay that is needed when TESLA is in use. This delay will make short lived path unavailable and so Ariadne will more often find the paths that are more stable and long lived.

Packet Overhead The packet overhead metric shows the number of routing and control packets sent.

Ariadne uses more packets than DSR but less than unoptimised DSR. This is also related to the more stable routes found by Ariadne than unoptimised DSR. This leads to fewer ROUTE ERROR packets sent, however, Ariadne tends to send more of these because of the delay in error processing and thus send more redundant packets.

Byte Overhead The byte overhead is the number of bytes sent that are not actual data. This metric is of course related to the packet overhead.

Since Ariadne has a larger packet header the byte overhead is higher in Ariadne than any of the other two protocols. However, in less dynamic topologies the byte overhead of Ariadne and unoptimised DSR tends to get closer.

Path Optimality Path optimality is a measure of the optimal route length, that is, the shortest possible with the given radio transmission range. The optimal path is computed off line and compared to the actual path found by the algorithms.

The DSR algorithm has an average of 0.6853 hops longer paths than the shortest available. Unoptimised DSR has additional 0.2705 hops and Ariadne

adds 0.0044 hops onto that. The explanation for unoptimised DSR being better than Ariadne is that the DSR algorithm sends out more route discoveries and thus tends to find shorter paths quicker.

Average Latency The average time between the packets get sent until they are received by the destination.

Ariadne tends to have lower latency than unoptimised DSR mainly because the lower number of broken links. Ariadne has an average latency below 0.8 seconds in highly dynamic topologies and going towards 0.5 seconds with more stationary nodes. Unoptimised DSR is above 0.9 seconds in all settings while DSR stays well below 0.2 seconds.

99.99th Percentile Latency This is just a statistical result based on the same latency measurement presented above.

Even these figures show that Ariadne generally has a shorter latency than the unoptimised DSR.

5.4.3 Authenticated Routing for Ad hoc Networks

The Authenticated Routing for Ad hoc Networks (ARAN) protocol is based on the assumption that every node has a valid public key to a trusted key server. Also, this key server must prior to use in the network sign each node's public key after secure authentication of the node. Sanzgiri et al. [27] leaves the details of this secure authentication to the developers and implementers of the system. Each certificate contains the node's IP address, the public key, the creation time, and an expiration time.

The source node that wants to contact a destination node broadcasts a route discovery packet containing the destination IP, the source node's certificate, a sequence number, and a time stamp. All this information is also signed using the node's private key. Each intermediate node that receives such a message checks the signature and if valid signs the data using its own private key and attaching the certificate of the node. If a node already has attached its certificate the next node in the path removes the certificate, signs the remaining data, and finally concatenates its own certificate in place of the previous one. For each valid route request an intermediate node sets up a reverse path back to the node from which it received the request.

When the request finally arrives at the destination it can validate the source. It can also choose from among different routes one which it seams most appropriate and unicast back a route reply, often the first route discovery packet is replied to. The chosen route may not be the shortest but the quickest. The route reply is signed by the destination node and the packet consists of the reply header, the IP address of the initial source node, the destination's certificate, the request sequence number, and the time stamp. The nodes along the path to the source verifies the previous

hop node's signature and certificate, removes them and signs the data using its own private key and concatenates its certificate.

During the communication links may fail due to movement or otherwise. In this case, when a node on the path discovers a down link it informs the initiator of this by sending out an error packet which it signs. The packet consists of the error header, IP addresses of both the source and the destination, its certificate, a sequence number, and a time stamp. This message is then forwarded toward the source node without further modification. The source cannot detect if this is a false alarm or not, but since the message is signed a malicious node cannot forge an error message for other nodes [27].

The protocol does not include any hop-count or source routing list. Instead it depends on each node along the path to keep the current routing information available. This is of course hard to get around except for using source routing lists. Additionally each node along the path must verify a signature and then sign data which does add some demands of computational power on the devices in the network, demands that may not always be possible to fulfil within a reasonable time. The need for a trusted key server might also be a single point of failure but using the ideas in section 5.3 this solution might be reasonable.

Performance Evaluation

To evaluate the ARAN protocol Sanzgiri et al. [27] have done simulations and comparisons to AODV. In their simulations they have measured six different metrics.

Packet Delivery Ratio Measures the ratio between the number of sent packets and the number of packets that reach the destination.

Just as in AODV the ARAN protocol has a very high packet delivery ratio, above 0.95 in all simulated situations.

Byte Overhead Ratio The routing control byte overhead in respect to received data bytes.

As many secure algorithms ARAN suffers from large overheads due to certificates and signatures stored in the packets.

Packet Overhead Ratio The ratio of routing control packets over data packets.

Simulations has shown that even though the byte overhead is larger in ARAN the actual number of control packets sent are roughly comparable to what is needed in AODV.

Average Path Length The path length is the number of hops each packet needs to take to reach its destination. The measure is the average of all path length in the simulation.

It was found that the average path lenght is very similar to the results of AODV. However, the authors note that it could be worse in case of heavier

data load where congestion could prevent ARAN to discover the shortest path.

Average Route Acquisition Latency This is the average time from the first sent route request until the corresponding route reply successfully arrived back to the initiator.

The average route acquisition latency in ARAN is nearly double that of AODV. This comes from the fact that each node needs to verify signatures on the way between the peer nodes and then calculate its own signature in addition to the regular packet processing which is done in AODV.

Average Latency This metric is a measure of the end-to-end data traffic latency once the route has been found.

The processing of data packets once a route is found is identical to both ARAN and AODV which the presented results show. The authors also note that the ratio between route acquisition and data packet delivery is often quite low why the higher acquisition latency will not matter at the end.

What would be interesting to see are some results showing the path length comparisons. The different algorithms should probably give somewhat different results in different settings in respect to node movement and topological dynamics. These differences should be clear in different mobility settings since the route discovery time is quite varying and thus some short lived routes would not be found if the route request process takes to long time.

5.4.4 Secure Ad Hoc On Demand Distance Vector Routing

The basic AODV protocol has built in capabilities for extension headers. The Secure Ad hoc Distance Vector (SAODV) protocol is a proposal by Zapata [36] for such extension headers. The extensions are used to send signatures and hash values that are later used for verification of the routing packets.

The SAODV is not meant to yield any confidentiality since this is usually not needed or desired in general ad hoc networks. The protocol does provide means to get authentication, integrity, and non-repudiation of the routing control packets.

The protocol extensions use asymmetric cryptography to achieve authentication by signing the data packets with the private key. This allows for the destination node, and all intermediate nodes, to validate the request. Also, this allows for the nodes to be certain that no one has altered the packets. However, some fields of the packets must change and these are signed as if they were zeroed out.

To allow for verification of the hop count field a one-way hash chain is utilised. The initiator of the route request decides a max hop count, such as ten or fifteen. It also generates a random value which is sent as the hash for the first hop count. The value is also hashed the max hop number of times producing a so called top hash. Each node can verify the hop count by checking that the incoming hash value

hashed max hop count minus the current hop count number of times is equal to the top hash. Since the top hash value is not changed, and thus signed, this provides the means to authenticate even the mutable hop count.

The SAODV extensions allow for two different ways for nodes to reply to a route request. The first way is to only allow the destination to reply. In this way the protocol works as described above. The destination node creates a route reply and signs it using its own secret key. The route reply is sent according to the usual AODV and each intermediate node can verify the reply and discard it if not valid. This approach does not consider the possibility of having intermediate nodes reply directly if they do have a valid route already.

To add the ability for route discovery optimisation a double signature scheme is devised. For each route request a second signature is added to the packet. This signature is stored in each intermediate node when they set up the reverse route. Later on, when a new route is needed because of node movement between the two peers an intermediate node that still has a route can reply directly by also including the second signature and the original signature. In addition to this, the actual life time is also sent in the reply which is signed by the intermediate node that sends the reply [36].

This method only allows malicious hosts to build a larger hop count or the same. That is, it is vulnerable to a partly replay attack which would let the hop count to be unchanged for one hop. This in turn could cause the route via the malicious node to be chosen as the shortest one. For this possibility to continue over time the adversary must be close to the actual shortest path and also move along while the intended peers move. Using more nodes in the attack would of course ease the attack at the cost of more attackers, which makes them easier to descry.

5.4.5 SEAD: Secure Efficient Ad hoc Distance Vector Routing

The secure efficient ad hoc distance vector (SEAD) routing protocol is a security aware modification of DSDV. SEAD is proposed by Hu, Johnson, and Perrig [8]. Most of the original operations of the DSDV are still used with some modifications. In addition, the distance vector updates are authenticated using an efficiently computable one-way hash function.

SEAD uses one-way hash chains to authenticate the distance vector updates from each node. These values are used in a way that disables malicious nodes to forge distance vector values with lower distance than the correct one. They are, however, free to add a larger distance if they see fit. In these situations the route is discarded in favour of a shorter path.

Each node creates a one-way hash chain that is later used for authentication of the node. The chain is split up into a number of parts each consisting of m subsequent entries. Given the hash chain

$$h_0, h_1, h_2, \ldots, h_n$$

where $m|n$. Each group of the chain with m elements corresponds to a sequence number, as is used in DSDV. The elements for a given sequence number i are

$$h_{(\frac{n}{m}-i)m}, h_{(\frac{n}{m}-i)m+1}, \ldots, h_{(\frac{n}{m}-i)m+m-1}.$$

The entry $h_{(\frac{n}{m}-i)m+j}$ for $0 \leq j < m$ is used to authenticate the entry with a distance of j hops. As in many other distance vector protocols the value m is to be considered infinite and there can not exist a path of any length greater or equal to m.

When a node sends out a distance vector update it adds the entry to itself with the distance metric equal to zero. Together with the routing table entries sent the hash value is also sent out. For each entry to other destinations it copies the current values it holds in the routing table and applies the hash function to the hash value it received for that entry.

A node that receives the distance update can easily authenticate the correctness of the shortest route by checking the hash value against the authenticated hash value for that node. In addition, the source of each routing update message must be authenticated. This can be accomplished using TESLA or some other broadcast authentication algorithm available [8].

6

Global IP Connectivity with Ad Hoc Networks

As the use of powerful laptops and mobile platforms increase the demand for connectivity grows as well. To support the mobile nature of people while still letting them stay connected is one area where ad hoc networks might come in handy. In addition, the users should not have to reconfigure and update their connection settings every time they move around to different Internet connections. To overcome some of these problems Mobile IP has been proposed. In this chapter the combination of Mobile IP (MIP) and ad hoc network technology will be investigated in a security aware way.

First a short description of the setup and possibilities with this system is presented. The security needs and problem statement follows. The main part of the chapter will then propose a security solution that is well fitted for the problem without causing too much overhead.

6.1 System Setup

The setup consists of three main parts. First of all, at the user end-point the local access network is a wireless network running AODV. The use of an ad hoc network as the access network gives a cheap and fast solution for service providers to extend connectivity and keeps the administrative burden as low as possible. These kinds of “hot-spots” are commonly popping up all over the place. First at places like airports, hotels, and other public places with a dense population of business related individuals. Currently even gas-stations are building “hot-spots” to support travelling sales-people and others, even pubs are building these connection points.

The “hot-spots” are linked together by some infrastructure giving access to the Internet. The access points are usually built up around wireless local area networks (WLAN) commonly using the 802.11 standard. The 802.11 does have some support for roaming between base stations if these are on the same network. The mobile host (MH) connection information are exchanged between the base stations using the infrastructure behind them. This kind of roaming is done on the data link layer and thus only works within the same sub network. The problem arises when the MH moves between different networks. The proposed solution to

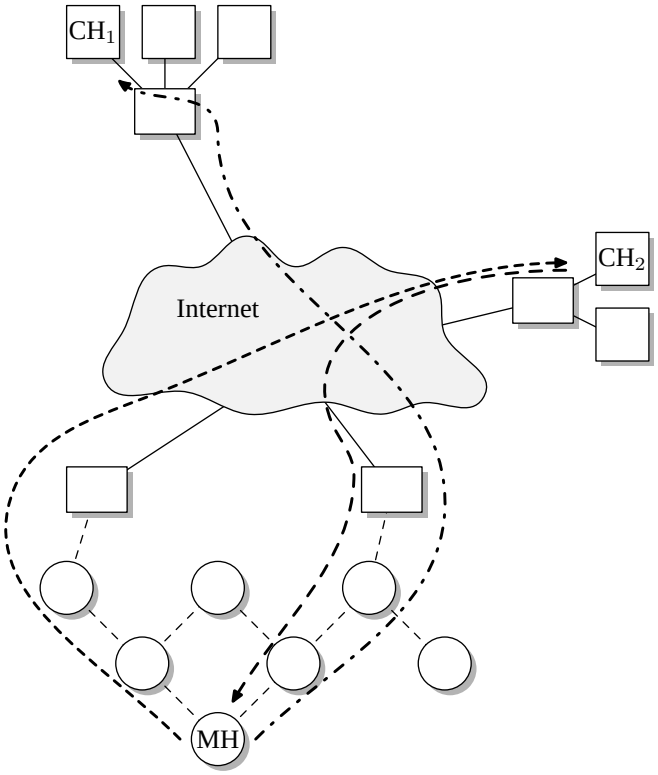

Figure 6.1. *Connectivity with MIP multihoming and AODV connection network. The thin dashed lines represent wireless links in the ad hoc connection network. The thicker dashed and dash-dotted lines are example routes for network traffic between the mobile host and a pair of corresponding hosts.*

this is to use Mobile IP (MIP). The MIP solution allows an MH to move around freely between networks without the need for any corresponding node to have any special support except for a standard IP stack.

The third part of the setup is the glue between the MIP that is mainly aimed at infrastructure networks and the local access network running an ad hoc network. This glue is implemented in a gateway preferably positioned within the base station. To further make all these parts work together more efficiently some modifications to the AODV and MIP protocols are proposed [40].

For MIP these changes involve multihoming, a technique that allows each MH to register with multiple care-of-addresses (COA). This is taken advantage of in route optimisation. In MIPv4 the home agent (HA) is responsible to choose the currently best available connection. This is accomplished using a metric based on deviation of round trip times between COA updates and acknowledgements. If MIPv6 is used the corresponding host (CH) is also active in this choice. The metric is calculated by the MH itself and reported back as part of multiple registrations, or binding updates in MIPv6 [39].

In the case of AODV some changes are made to accommodate the MIP information. Also, to make things more efficient, agent advertisements that are broadcast within the ad hoc network are used to set up initial routes to the agent from each node. This way an extra route request (RREQ) is spared [38].

6.2 Routing and Node Discovery

In a general IP network the routing is based on hierarchical use of the address space. Each host on the network must have a topologically correct address to be able to engage in communication with other hosts. In the AODV protocol, which is used for the ad hoc network, the address space is flat. That is, there is nothing in the address, such as network number, that tells how to find a specific host[1]. Instead a reactive protocol, much in the same sense as the Address Resolution Protocol (ARP) is used in wired Ethernet, is used to find the route to a specific host. For an overview description of the AODV protocol see section 2.3.2 on page 10 or refer to [21] for a full description.

When a MH needs to connect to a service on the Internet it sends out a RREQ as usual to find a route to the host providing the service. The RREQ traverses the ad hoc network in, possibly, multiple hops until it reaches the gateway. The gateway concludes that the host asked for is not inside the ad hoc network and therefore acts as a proxy. The gateway returns a route reply (RREP) thus setting up a route to itself. When this RREP has reached the MH it starts sending data packets using this route. The gateway forwards them onto the wired network and from then on regular IP routing is in action.

To allow the gateway to know whether a given host is on the Internet or in the ad hoc network it closely works together with the foreign agent (FA) which registers all the visiting hosts. If a host has the same network number as the ad hoc network it is assumed that the node is within the wireless network. Otherwise a lookup in the visitor list is done to check if the node is visiting or not. If the node is actually on the ad hoc network the gateway acts just like any other AODV node and forwards the RREQ to let the actual destination reply.

If the ad hoc network uses several gateways and access points (AP) these are needed to be synchronised. The synchronisation is done over the wired network to spare the scarce resources in the wireless media. For every addition and removal of visiting nodes the gateways needs to synchronise. Otherwise a moving MH might end up using one AP but was initially registered at another. Without the synchronisation the current AP will not detect the node in its visiting list and conclude that it is not on the wireless network. This also allows the MH to register different COAes with different FAs.

[1] There is, however, the ability to specify “net masks” in the route reply message thereby creating subnets within the otherwise flat address space [21].

6.3 Problem Statement

As is the case in all application the peer-to-peer security is solved using existing end-to-end cryptographic protocols. That is, all the application data that is important to secure should be sent encrypted on the higher levels. The problem is securing the routing in the ad hoc network and the COA registration at the HA and CH in a safe way. For this discussion the security aspects of MIP is not heavily considered. The security issues in MIPv4 can be solved using tunnelling and for MIPv6 with the support of Authentication Headers (AH) and Encrypted Security Payload (ESP) the security goals can be reached.

Since the application of these protocols are mainly targeted at public "hot-spots" there is no need to keep routing information secret. The need for security of the protocols is mainly to try to keep the services available and the routing information trustworthy. For this the integrity of the routing information is the main task in the ad hoc network. For the rest of the communication path the current Internet routing could be considered good enough. However, the authentication of nodes must be addressed.

In certain situations authorisation to use certain foreign networks could also be an issue. In this case, different solution based on certificates using for example SPKI [5] could be investigated.

6.4 Securing Mobile IP

To get MIP secure there are some major concerns relating to binding updates and COA. These should be simple to address using signatures for authentication between HA and MH. In MIPv6 IPSec can be used by all parties. In MIPv4 the only general solution[2] is to not use route optimisation and thereby always using tunnels between the MH and HA in both directions as is proposed in [17].

In the solutions a key must be shared between the parties. In the case of tunnelling between MH and HA this should not be such a big problem since at setup and when "at home" the key exchange can be done easily either manually or in some automatic way. The other parties in the protocol must be authorised as well. Since they all are connected to the Internet it is possible to use any on line key service. To not be bound to a single point of failure a PGP like structure could be used. The public key certificates can thus be fetched on demand and cached at each node for later use. The caching must not interfere with any revocation of keys or certificate why this should be controlled frequently enough.

In addition, the actual key that is used for communication should be decided upon for each session and exchanged at certain intervals. This further minimises the risk of replay attacks. Also, an attack of a single session should not jeopardise future exchanges.

[2] A solution that is not intruding on current implementation of the IP stack at CH and intermediate nodes.

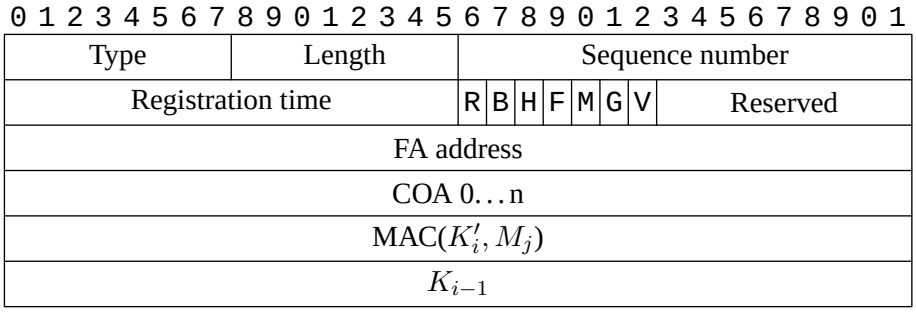

Figure 6.2. *The modified agent advertisement mobility extension for the secure ad hoc network. The MAC is calculated for interval i using K_i' on the current message M_j. At the same time, disclose the key K_{i-1} used in the previous interval.*

6.5 Securing the Ad Hoc Network

6.5.1 Foreign Network Discovery

For a MH to detect an available foreign network the FA sends out agent advertisements. In a security aware setting the MHs must be certain that the agent advertisement is from the real FA since the advertisements are used to set up routing paths toward it. To accomplish this my proposal is to use TESLA. For this to work there are some changes needed in the agent advertisement packet which is depicted in figure 6.2. The agent advertisement package is a mobility extension to the RFC 1256 [3] ICMP router advertisement message. The changes incorporate a Message Authentication Code (MAC) and a TESLA interval key. By letting the gateway append a MAC to the message each node in the wireless network can verify the packet and be certain that it is the real FA. The use of TESLA provides for an efficient implementation since only symmetric cryptological primitives are in use.

There are, however, a few things that must be noted. The use of TESLA demands loose time synchronisation between the sender and the receivers. Each MH must be able to tell if a certain message was received within a certain key disposal interval. With current stable time circuits and time synchronisation protocols this should not be an issue. The MH can synchronise using for example Network Time Protocol (NTP) when at their trusted home network or connected in any other way.

The largest problem in the case of using TESLA for the authentication is that the MH must have an authenticated disclosed key for the current key chain. One way would be to append a certificate and a signed key to the agent advertisement packet. This is not a satisfactory solution since the information is only needed once, when the node enters the network. Instead the nodes currently in the network can supply the key either on request by the new MH or on detection of a new neighbour. This solution will result in new control packets but otherwise keep the protocols unchanged. See section 6.5.3 for further details about key management and certificate distribution.

Another topic that must be solved is that of the key disclosure scheme. I suggest that each agent advertisement sent out starts a new interval in the scheme. Since the agent advertisement gets sent out regularly and the intervals are relatively long, at least 3 seconds, the time is not too short to reach any node available in the network. The actually used key disclosure scheme should be added as an extension to the signed certificate.

Forward Path Setup

The verified and authenticated agent advertisement message is used to set up the routing table. The previous sender is added as the next hop toward the gateway. The metric, M, is set using the deviation in arrival times of the agent advertisement messages according to equation 6.1 [38, 40]. This way there is no need to secure any hop count since the computations are done locally in each host.

$$\begin{aligned} T_\Delta &= T_{\text{Current}} - T_{\text{Last}} \\ \bar{\Delta} &= T_\Delta \times \delta - \bar{\Delta} \times (1 - \delta) \\ M &= (T_\Delta - \bar{\Delta})^2 \times \mu + M \times (1 - \mu) \end{aligned} \tag{6.1}$$

The constants δ and μ are configurable to give a metric that is as true as possible in the actual setup. Åhlund and Zaslavsky [40] are set out to further study the selection of these parameters.

The use of this technique depends of multiple receptions of agent advertisements since the metric is calculated from the measured time between arrivals. Also, to be able to verify an advertisement the MH must wait for the next advertisement message to arrive.

As each node has an authenticated route to the gateway the forward path setup and gateway authentication is complete.

6.5.2 Registration

As the MH has gained knowledge of the foreign network and the FA is authenticated the MH has to register with the FA and the HA. The registration is done by sending a registration message to the FA that is forwarded to the HA. In the registration message the home address, home agent, and the chosen COA is included together with a set of flags.

A route is already setup towards the FA that can be used directly. The security of the MIP registration is left for higher layers or simply using IPSec as mentioned above. When the reply is going to get sent back the gateway needs to find a route securely to the MH. This is accomplished using the SAODV as in all other routing within the ad hoc network.

During further networking and maybe detection of additional gateways the MH can choose to register for multiple COAes with different FAs. The HA or

CH choose the one with the best metric. To find out what each COA route metric is the MH measures the round trip time (RTT) between a registration message sent and the corresponding reply. The metric is then calculated according to equation 6.2 [40] which is also used for the retransmission timer in the TCP protocol [34, pp. 539–542].

$$\begin{aligned} \Delta &= \mathrm{RTT} - \mathrm{EstimatedRTT} \\ \mathrm{EstimatedRTT} &= \mathrm{EstimatedRTT} + (\delta \times \Delta) \\ \mathrm{Deviation} &= \mathrm{Deviation} + \delta(|\Delta| - \mathrm{Deviation}) \\ M &= \mu \times \mathrm{EstimatedRTT} + \phi \times \mathrm{Deviation} \end{aligned} \tag{6.2}$$

This metric is sent along with the next registration message. This way the CH and HA knows about the metric for each different COA since those could be routed in different ways. The metrics also help to choose the path with best connectivity further increasing availability and better service. Paths with malicious nodes that delay data packets are automatically rejected due to these calculations thus providing for a more stable connection.

6.5.3 Key Management

To secure the routing within the ad hoc network the security extensions to AODV as proposed by Zapata [36] are used. The extension is basically a signature of the message and a hash-value used in a hash chain to secure the hop count. For the protocol to work as expected each node must be able to verify the signatures which is the main problem in this setup. Also, the verification process of the agent advertisement is also in need of a certificate of authenticity.

To be able to verify a signature the verifying node must know the public key of the originating host[3]. The key can simply be sent by the originator along with the data packet. However, this will not be very secure since just about anyone can send out a key claiming to belong to someone else. What is needed is authentication.

As always, this is usually provided with the help of certificates. But a certificate itself is often relatively large and does not fit well to bee sent around in each routing packet. Instead an ad hoc protocol is needed to get the certificate of a specified host.

In this case I propose to use a limited broadcast to ask any neighbour for the valid certificate. The assumption is that each node has a certificate containing its own public key and that the certificate is signed by some CA. I also assume that the CA is a trusted third party (TTP) for all nodes in the network, either explicit or implicit.

When a node needs to find a specific certificate for verification of routing packets it can send out a limited broadcast. That is, a broadcast with a time to live set to one. This way it will not pollute the entire network. Since the request is only used for getting the certificate for routing packets it is guaranteed that at least one

[3] Unless a symmetric key scheme is used in which case the nodes must exchange the shared secret somehow in advance.

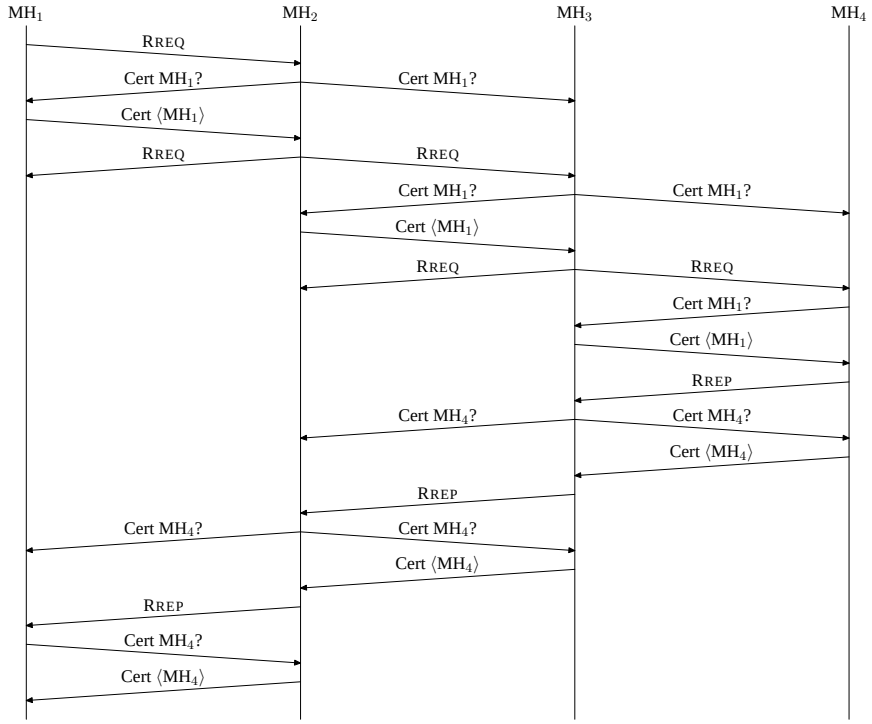

Figure 6.3. *Route discovery with SAODV and on demand distribution of certificates. MH_1 is trying to find a route to MH_4.*

of the neighbours already knows about this certificate and can forward it to the one in need.

The process of finding a route between two hosts in an ad hoc network is exemplified in figure 6.3. In this example the MH_1 is trying to reach MH_4 in a simple network. Broadcasts are shown as messages going to more than one destination.

To make the route requests more efficient in the future each node should cache the certificates. This way a certificate may only be asked for once for each node. The size of the cache should be large enough to hold a reasonable amount of certificates for proper operation. The number of certificates could be managed by a simple cache algorithm that throws out the least recently used certificate if the cache space is limited.

6.5.4 Limitations

There are still problems with malicious nodes that can use a grey-hole attack in case they forward the agent advertisement messages but not other packets. When this happens the route is detected as unavailable due to time-outs and a regular route request is initiated. This will not solve the problem, however, it should be possible for a neighbouring node to detect the malicious node and raise the alarm.

The use of TESLA authenticate injects a delay from the reception of a message until it can be verified which leads to an extra buffering of packets. This also

means that the metric calculations done on the arrival still can be done at once, but to update the actual metric must be postponed until later when the message is verified.

Certificate revocations are not supported until a node has a valid route toward the Internet and an on line CA service. The problem is kind of a chicken and egg problem. Without signatures and certificates we can not provide for a secure route, but without a route to the on line CA we can not be certain that the certificate is actually valid.

In the proposed solution there is always a need to register with the gateway/FA since it might need to synchronise with other AP serving the same sub network. This means that there is still need for a FA even in MIPv6.

7

Conclusions and Further Work

In this master's thesis I have dug into different cryptological and ad hoc routing algorithms. There are a plethora of proposed solutions but none that really can be seen as a general solution. In this chapter a short discussion about what to choose in different situations will follow. Some open issues and pointers for further investigation will be addressed in section 7.2.

7.1 The Appropriate Choice

As stated above and as can be seen in chapters 2 and 5 there are a lot of different algorithms to achieve ad hoc routing. With the anticipated growth of connected devices the usage of ad hoc networks will probably increase. As more and more uses for these devices pop up I am certain that the need for securing the communications will be a high priority in the future. For wireless transmission there is no physical safety that can be built up around the trunks of cables and routers. This leads to a higher need to make sure the lower layer network protocols are intact.

There are solutions proposed but all suffers from some drawbacks, mainly related to key management or efficiency in computations. The only general conclusion I can see is that it all depends on the situation at hand.

7.1.1 Cryptography

To secure end-to-end traffic the recommendation falls on hybrid algorithms in which asymmetric cryptography is used to set up the session and the bulk traffic is encrypted using more efficient symmetric primitives. This way all kinds of hosts can be used and a handshake will make sure the best available solution is used in each case. This is how SSL and TLS are working today and they have gained very much popularity.

As to what specific algorithms to use I propose the use of elliptic curve systems for asymmetric encryption for its higher efficiency in relation to other public key systems. Since the modulus can be kept smaller the computations will be smaller and thus faster.

In the case of symmetric systems the standard AES will be the most widely spread and has undergone much review. There also exist many open source imple-

mentations and the algorithm was from the very beginning designed to be efficient in a wide range of hosts.

7.1.2 Routing

For a general setting I would recommend the AODV protocol and for a security aware setting the SAODV extension. The AODV protocol has been around for quite some time and is constantly being reviewed in the MANET working group at IEEE. Most simulations done on other protocols compare against AODV and often AODV seems to be the most efficient choice.

In special setups the usage of other routing protocols might be better. For example in networks where there are few bidirectional links a source routing protocol such as DSR can handle different routes in different directions. These kinds of protocols are also somewhat easier to secure in relation to hop count based protocols. This is because of the higher degree of control the peers and also the intermediate nodes have.

7.1.3 Key Management

The issues of key management are the hardest and I have found no satisfactory solution to this problem. The actual problem as I see it is the chicken and egg situation in which it is impossible to secure the routes without authentication done in some way. On the other hand, to be able to authenticate we need the actual routes set up.

I proposed a simple on demand protocol for sharing of certificates. The main problem is the problem of certificate revocations. The kind of certificate does not matter in the solution, the only assumption is that the verifying node is able to verify the certificate without needing to get even more certificates from some arbitrary service. This poses, as usual, the need for a common trusted entity or possibly setting up the certificates prior to entering the ad hoc network.

In the setting discussed in chapter 6 the ability to later contact an on line CA or key server for key management is something that is not available in all settings. But as the connectivity to the Internet gets everywhere this might be as good as it needs to be. By first setting up a route using a on demand shared certificate and later, perhaps using previously shared secrets, control certificates and accomplish other key management tasks in a secure way with a trusted on line service the security might be enough. Even if the first shared certificate is spoofed, the traffic to the on line service can not be, and if that connectivity can not be given by the setup route the exposed node will detect the fraud.

For more simple networks such as remote controls to the home entertainment system a solution in the spirit of the resurrecting duckling fits well into place. It is simple, has low demands on hardware and can be secure enough. Here key management is as simple as holding the two devices together and exchange the secret.

7.2 Further Work

There is always a trade off between efficiency and security in the general case. The cost of different security aware solutions must be further studied and put into relation to the information value that they can protect. With such studies the selection of the most appropriate algorithms will be much easier. Some metrics that also might be interesting are of course the impact of the larger routing packets, lengthier packet processing times, and overall route set up time.

The impact of different attacks would also be an interesting field to further investigate. Simulation models and prototypes will be very important as well as comparable attacker models that can be used. However, each protocol has its own weaknesses why certain attacks might only work on certain algorithms. A general taxonomy of attacker models and attacks are needed.

In some of the proposed solutions for secure ad hoc routing there are still some known attacks. For example, in all distance vector based algorithms that use hash chains for securing the hop count an adversary is able to replay the previous hop count and hash value. Are there any elegant and efficient solutions to this problem? Is it really a problem? A simulation with some fraction of adversaries doing this kind of attack should show how large the problems really are.

The ultimate testbed would be a prototype setup where the metrics and statistics can be measured for real life node movements and usage. The problem is that of generality as it is hard to set up an experiment that can be said to be general. For the idea of the extension of Internet connectivity using ad hoc networks the usage pattern will be much the same in most “hot spots” why a setup for this will prove very useful. The current setup would have to be extended to take into account the security measures that are proposed.

For the proposed on demand distribution of certificates in general ad hoc networks some metrics will be most interesting to find out. First of all is the memory needs for each node. How many certificates will each node have to cache for proper functionality of the routing? This is heavily dependent on the traffic patterns across the network and the number of routes going through a certain node. Since the usage is mainly aimed at extending the last hop most traffic will probably go through the gateway towards the Internet. This will mean that the nodes closest to the gateway will have more routes to handle than other nodes in the network. How will this affect the actual usability for these nodes?

Glossary

AES	Advanced Encryption Standard, a new encryption algorithm that are to replace the old DES. It is more secure and can be efficiently implemented in various hardware architectures. AES is based on the Rijndael encryption algorithm.
AH	Authentication Header, a special header used in IP Security for authentication purposes.
AODV	Ad hoc On demand Distance Vector, a routing protocol for ad hoc networks based on DSDV.
AP	Access Point, a base station or other place for connection to an available infrastructure.
ARAN	Authenticated Routing for Ad hoc Networks, a security aware routing protocol.
ARP	Address Resolution Protocol, a protocol used in LANs for detection of multiple access control addresses.
CA	Certification Authority, a trusted authority that hand out certificate by signing them with its private key. CAs are used in PKI solutions and every one involved must trust this single service for the system to work.
CBC	Cipher Block Chaining, a mode of operation for block ciphers. In this mode each ciphertext block that is output is x-ored with the next plaintext block before that block is encrypted.
CFB	Cipher Feedback, a mode of operation for block ciphers where a key stream is generated by encrypting the previous output and x-oring this with the plaintext to achieve the next output value.
CH	Corresponding Host, a host that a mobile host is communicating with. The term is commonly used when talking about MIP.
COA	Care Of Address, the address used by a mobile node visiting a foreign network.
DES	Data Encryption Standard, a old encryption and decryption algorithm using shared secret keys. The algorithm was based on the IBM Lucifer algorithm.
DHCP	Dynamic Host Configuration Protocol, a protocol that allows for dynamic configuration of hosts in a network. The configuration items

might be such things as the IP address, default gateway, and domain name server addresses.

DSDV Destination Sequence Distance Vector, an early protocol for routing in ad hoc networks.

DSR Dynamic Source Routing, a routing protocol for ad hoc networks.

ECB Electronic Codebook, a mode of operation for block ciphers. This is the usual usage of a block cipher where each input block is encrypted using the same key and all blocks are treated independent of each other.

ECC Elliptic Curve Cryptosystem, a cryptosystem that is based on the mathematical group of discreet values on a specified elliptic curve.

ESP Encryption Security Payload, an extension header used in IP Security for encryption of data traffic.

FA Foreign Agent, an agent that are part of the MIP extension to IP. It handles registration and helps the MH with different tasks involving detection of available networks.

GF Galois Field, a mathematical structure.

HA Home Agent, an agent that is available in a MH's home network. The HA can redirect and forward traffic to the MH when the MH is not at home. This is used in MIP.

ICMP Internet Control Message Protocol, a protocol used in the IP network for network management and control. Example of ICMP message are the widely used PING packet that should be echoed to verify available hosts. Another example is the router advertisement message.

ID Identity, the common name or number that an individual or computer host are identified by.

IPSec Internet Protocol Security, extensions to the IP protocol that allows for security aware network applications.

IP Internet Protocol, the network protocol that is used for addressing on the Internet.

IV Initialisation Vector, usually used in encryption routines where a start value is needed to seed of a stream cipher or a block cipher used in a stream setting.

LAN Local Area Network, a network used within a restricted area for local communication. Several such networks can be connected forming larger systems.

LFSR Linear Feedback Shift Register, a commonly used hardware structure for building PRNGs. It consists of shift registers with feedback lines to insert the new bits depending on the currently set values in some of the bits.

MAC Message Authentication Code, a cryptographic secure code that identifies and authenticate the contents of a message.

MANET Mobile Ad hoc Network, a network that consists of mobile nodes that spontaneously can connect to each other without help from any preexisting infrastructure.

MD4 Message Digest algorithm version 4, a fast and efficient but not very secure hash algorithm useful for computing digest codes from messages.

MD5 Message Digest algorithm version 5, a hash algorithm largely based on MD4. It is a little slower but more secure.

MDC Modification Detection Code, a message signature useful for detection of changes in data sent over non error-free channels.

MH Mobile Host, a mobile host in a network.

MIP Mobile IP, mobility extensions for supporting roaming in IP networks.

ND Neighbour Discovery, a protocol for detection of neighbours in a LAN. This is part of IP version 6.

NTP Network Time Protocol, a protocol used to keep on line computers synchronised. The protocol uses advanced time measurements and statistics to achieve as good result as possible.

OFB Output Feedback, a mode of operation of block ciphers where they are used for generating key streams in a stream cipher kind of way.

PAN Personal Area Network, a network consisting of multiple small devices that are usually carried by a single individual and makes up for his/her personal information needs.

PGP Pretty Good Privacy, an open architecture for certificate handling based on a distributed trust model.

PKG Private Key Generation, a service that is able to generate private keys given the ID of the host. This kind of service can also be distributed over many participants to get a higher degree of security and availability.

PKI Public Key Infrastructure, an idea of how to handle certificate in an efficient and trustworthy way.

PRNG Pseudo Random Number Generator, an algorithm that outputs seemingly random numbers but that can bee seeded and controlled to output the same sequence multiple times.

RERR Route Error, a control packet used in many ad hoc network protocols to signal a link break in a previously set up route.

RFC Request For Comment, a document used in many Internet working groups to propose different standards. The documents are often referred to as the standards used in the Internet.

RREP Route Reply, a control packet used in many on demand protocols in the set up phase for routes in the network.

RREQ Route Request, a control packet used in many on demand protocols in an initial phase when trying to set up a route in the network.

RSA Rivest, Shamir, and Adleman are the inventors of the RSA cryptosystem that was the first public key system realised.

RTT Round Trip Time, the time from a message is sent until a reply has been received again.

SAODV Secure AODV, a proposal for extension headers to build a more secure variant of the AODV protocol.

SEAD Secure Efficient Ad hoc Distance vector protocol, a security aware routing protocol for ad hoc networks.

SHA-1 Secure Hash Algorithm first revision, a cryptological secure hash function with a 160-bit output value.

SPKI Simple Public Key Infrastructure, a standard used for allowing the use of certificate chains.

SRP Secure Routing Protocol, a security aware ad hoc routing protocol.

SSL Secure Socket Layer, a protocol used for negotiation of encryption algorithms and secure transmission over any end to end connection.

SUCV Statistically Unique Cryptographically Verifiable, algorithms to randomly generate unique identifications that are verifiable by cryptographic means.

TCP Transmission Control Protocol, a protocol for handling connection oriented traffic above the IP layer.

TESLA Timed Efficient Stream Loss-tolerant Authentication, a protocol for authentication of broadcast data in an efficient way.

TLS Transport Layer Security, a protocol used for negotiation of encryption algorithms and secure transmission over any end to end connection. This is a standardised version that are based on SSL.

TTP Trusted Third Party, an independent third party that is trusted by communicating parties.

WLAN Wireless Local Area Network, a local area network using a wireless transmission technique to communicate over the air.

ZRP Zone Routing Protocol, a routing protocol suited for large ad hoc networks which is a hybrid of different routing protocols used in different settings.

References

[1] COULOURIS, G., DOLLIMORE, J., AND KINDBERG, T. *Distibuted Systems Concepts and Design*, third ed. Pearson Education, 2001. ISBN: 0201-61918-0.

[2] DAEMEN, J., AND RIJMEN, V. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002. ISBN: 3-540-42580-2.

[3] DEERING, S. E. ICMP Router Discovery Messages. RFC 1256, Sept. 1991.

[4] ELLISON, C., AND SCHNEIER, B. Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure. *Computer Security Journal XVI*, 1 (2000), 7.

[5] ELLISON, C. M., FRANTZ, B., LAMPSON, B., RIVEST, R., AND AHDN TATU YLONEN, B. T. SPKI Certificate Theory. RFC 2693, Sept. 1999. Experimental.

[6] GORDON, J. The Alice and Bob After-Dinner Speech. Webpage, 1984. http://www.conceptlabs.co.uk/alicebob.html.

[7] HAAS, Z. J., AND PEARLMAN, M. R. *ZRP A Hybrid Framework for Routing in Ad Hoc Networks*. In Perkins [20], 2001, ch. 7, pp. 221–253.

[8] HU, Y.-C., JOHNSON, D. B., AND PERRIG, A. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. Tech. rep., Rice University, University of California, Berkeley, 2002.

[9] HU, Y.-C., PERRIG, A., AND JOHNSON, D. B. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *Eigth ACM International Conference on Mobile Computing and Networking (Mobicom 2002)* (Sept. 2002).

[10] HUBAUX, J.-P., BUTTYÁN, L., AND ČAPKUN, S. The Quest for Security in Mobile Ad Hoc Networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing* (2001).

[11] JOHNSON, D. B., MALTZ, D. A., AND BROCH, J. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*. In Perkins [20], 2001, ch. 5, pp. 139–172.

[12] JONES, G. A., AND JONES, J. M. *Elementary Number Theory*. Springer-Verlag, 1999. ISBN: 3-540-76197-7.

[13] KHALILI, A., KATZ, J., AND ARBAUGH, W. A. Toward Secure Key Distribution in Truly Ad-Hoc Networks. Tech. rep., University of Maryland (College Park), 2003.

[14] KUROSE, J. F., AND ROSS, K. W. *Computer Networking: A Top–Down Approach Featuring the Internet*. Addison–Wesley Publishing Company, 2001. ISBN: 0-201-47711-4.

[15] LUNDBERG, J. Routing Security in Ad Hoc Networks. Tech. Rep. Tik-110.501, Helsinki University of Technology, 2000.

[16] MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. *Handbook of Applied Cryptography*. CRC Press LLC, 1997. ISBN: 0-8493-8523-7.

[17] MONTENEGRO, G. E. Reverse Tunneling for Mobile IP. RFC 2344, May 1998.

[18] NICHOLS, R. K., AND LEKKAS, P. C. *Wireless Security Models, Threats, and Solutions*. McGraw–Hill, 2002. ISBN: 0-07-138038-8.

[19] PAPADIMITRATOS, P., AND HAAS, Z. J. Secure Routing for Mobile Ad hoc Networks. In *SCS Communication Networks and Distributed System Modeling and Simulation Conference* (Jan. 2002).

[20] PERKINS, C. E., Ed. *Ad Hoc Networking*. Addison–Wesley Publishing Company, 2001. ISBN: 0-201-30976-9.

[21] PERKINS, C. E., BELDING-ROYER, E. M., AND CHAKERES, I. D. Ad hoc On-Demand Distance Vector (AODV) Routing. IETF Internet-Draft, Oct. 19, 2003. `draft-perkins-manet-aodvbis-00.txt`, work in progress.

[22] PERKINS, C. E., AND BHAGWAT, P. *DSDV Routing over a Multihop Wireless Network of Mobile Computers*. In Perkins [20], 2001, ch. 3, pp. 53–74.

[23] PERKINS, C. E., AND ROYER, E. M. *The Ad Hoc On-Demand Distance-Vector Protocol*. In Perkins [20], 2001, ch. 6, pp. 173–219.

[24] PERRIG, A., CANETTI, R., TYGAR, J. D., AND SONG, D. The TESLA Broadcast Authentication Protocol. Tech. rep., UC Berkley, IBM Research, 2002.

[25] RIVEST, R. L. The MD4 Message-Digest Algorithm. RFC 1320, Apr. 1992. Obsoletes RFC 1186.

[26] RIVEST, R. L. The MD5 Message-Digest Algorithm. RFC 1321, Apr. 1992.

[27] SANZGIRI, K., DAHILL, B., LEVINE, B. N., SHIELDS, C., AND BELDING-ROYER, E. M. A Secure Routing Protocol for Ad Hoc Networks. Tech. Rep. UM-CS-2001-037, University of California, Santa Barbara; University of Massachusetts, Amherst; Georgetown University, Washington, 2001.

[28] SCHILLER, J. *Mobile Communications*. Addison–Wesley Publishing Company, 2000. ISBN: 0-201-39836-2.

[29] SCHNEIER, B. *Applied Cryptography Protocols, Algorithms, and Source Code in C*, second ed. John Wiley & Sons, Inc., 1996. ISBN: 0-471-11709-9.

[30] STAJANO, F., AND ANDERSON, R. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols, 7th International Workshop Proceedings* (1999), B. Christianson, B. Crisp, and M. Roe, Eds., Springer-Verlag Berlin Heidelberg. Lecture Notes in Computer Science.

[31] STAJANO, F. The Resurrecting Duckling — what next? In *Security Protocols, 8th International Workshop Proceedings* (2000), B. Christianson, B. Crisp, and M. Roe, Eds., Springer-Verlag Berlin Heidelberg. Lecture Notes in Computer Science, extension to [30].

[32] STEENSTRUP, M. *Cluster-Based Networks*. In Perkins [20], 2001, ch. 4, pp. 75–138.

[33] STINSON, D. R. *Cryptography Theory and Practice*. CRC Press LLC, 1995. ISBN: 0-8493-8521-0.

[34] TANENBAUM, A. S. *Computer Networks*, third ed. Prentice Hall, Inc., 1996. ISBN: 0-13-394248-1.

[35] WEIMERSKIRCH, A., AND THONET, G. A Distributed Light-Weight Authentication Model for Ad-hoc Networks. Tech. rep., Accenture Technology Labs, 2001.

[36] ZAPATA, M. G. Secure Ad hoc On-Demand Distance Vector (SAODV) Routing. IETF Internet-Draft, Oct. 10, 2001. `draft-guerrero-manet-saodv-00.txt`, work in progress.

[37] ZHOU, L., AND HAAS, Z. J. Securing Ad Hoc Networks. *IEEE Network 13*, 6 (1999), 24–30.

[38] ÅHLUND, C., AND ZASLAVSKY, A. Software Solutions to Internet Connectivity in Mobile Ad Hoc Networks. In *4th International Conference on Product Focused Software Process Improvement PROFES* (2002), Springer-Verlag. Lecture Notes in Computer Science.

[39] ÅHLUND, C., AND ZASLAVSKY, A. Multihoming with Mobile IP. In *6th IEEE International Conference on High Speed Networks and Multimedia Communications HSNMC'03* (2003), Springer-Verlag. Lecture Notes in Computer Science.

[40] ÅHLUND, C., AND ZASLAVSKY, A. Extending Global IP Connectivity for Ad Hoc Networks. *Telecommunication Systems 24*, 2-4 (2003), 221–250.